Introduction

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

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


Related documents

- STM32H563/H573 and STM32H562 datasheets
- STM32H563/H573 and STM32H562 errata sheet
Contents

1 Documentation conventions ............................................. 101
  1.1 General information ............................................. 101
  1.2 List of abbreviations for registers ............................. 101
  1.3 Glossary ......................................................... 102

2 Memory and bus architecture ........................................... 103
  2.1 System architecture ............................................. 103
    2.1.1 Fast C-bus ................................................ 104
    2.1.2 Slow C-bus ............................................... 104
    2.1.3 S-bus ..................................................... 104
    2.1.4 DCache S-bus ............................................. 105
    2.1.5 GPDMA1 and GPDMA2 buses ................................ 105
    2.1.6 SDMMC1 and SDMMC2 controllers DMA buses ............... 105
    2.1.7 BusMatrix ............................................... 105
    2.1.8 AHB/APB bridges ......................................... 105
    2.1.9 Ethernet MAC ............................................. 105
  2.2 TrustZone security architecture ................................. 106
    2.2.1 Default TrustZone security state .......................... 106
    2.2.2 TrustZone peripheral classification ....................... 107
  2.3 Memory organization ............................................. 111
    2.3.1 Introduction ............................................. 111
    2.3.2 Memory map and register boundary addresses ............... 112
    2.3.3 Embedded SRAM ........................................... 118
    2.3.4 Flash memory overview ................................... 118
    2.3.5 Boot modes .............................................. 118

3 System security .......................................................... 120
  3.1 Key security features ........................................... 120
  3.2 Secure install .................................................. 121
  3.3 Secure boot ..................................................... 121
    3.3.1 Unique boot entry ....................................... 122
    3.3.2 Immutable root of trust in system flash memory .......... 123
  3.4 Secure update .................................................. 123
### Contents

3.5 Resource isolation using hide protect levels ........................................ 123
3.6 Resource isolation using TrustZone .................................................. 124
  3.6.1 TrustZone security architecture .................................................. 124
  3.6.2 Armv8-M security extension of Cortex-M33 ................................ 125
  3.6.3 Memory and peripheral allocation using IDAU/SAU ......................... 125
  3.6.4 Memory and peripheral allocation using GTZC ............................ 127
  3.6.5 Managing security in TrustZone-aware peripherals ....................... 131
  3.6.6 Activating TrustZone security ................................................ 137
  3.6.7 Deactivating TrustZone security ............................................. 138
3.7 Other resource isolation ......................................................... 138
  3.7.1 Temporal isolation using secure hide protection (HDP) .................. 138
  3.7.2 Resource isolation using Cortex privileged mode ......................... 139
3.8 Secure execution ................................................................. 143
  3.8.1 Memory protection unit (MPU) ................................................ 143
  3.8.2 Embedded flash memory write protection .................................. 144
  3.8.3 Tamper detection and response .............................................. 144
3.9 Secure storage ................................................................. 146
  3.9.1 Hardware secret key management .............................................. 147
  3.9.2 Unique ID ............................................................................ 148
3.10 Crypto engines ................................................................. 148
  3.10.1 Crypto engines features ......................................................... 148
  3.10.2 Secure AES co-processor (SAES) ............................................. 149
  3.10.3 On-the-fly decryption engine (OTFDEC) .................................... 150
3.11 Product life-cycle ............................................................... 150
  3.11.1 Product configurations and security services ............................... 151
  3.11.2 Life-cycle management ......................................................... 152
  3.11.3 Recommended product settings ............................................... 155
3.12 Software intellectual property protection and collaborative development 155
  3.12.1 Software intellectual property protection ................................. 157
  3.12.2 Software intellectual property protection with OTFDEC .............. 157
  3.12.3 Other software intellectual property protections ....................... 159

4 Boot modes ................................................................. 160
  4.1 STM32H562/H563 boot modes ..................................................... 161
  4.2 STM32H573x boot modes ......................................................... 162
5 Global TrustZone® controller (GTZC) ........................................ 163
  5.1 Introduction ............................................................. 163
  5.2 GTZC main features ................................................... 163
  5.3 GTZC implementation .................................................. 165
  5.4 GTZC functional description ......................................... 166
    5.4.1 GTZC block diagram ............................................. 166
    5.4.2 Illegal access definition ....................................... 167
    5.4.3 TrustZone security controller (TZSC) ......................... 168
    5.4.4 Memory protection controller - block based (MPCBB) ....... 170
    5.4.5 TrustZone illegal access controller (TZIC) .................. 170
    5.4.6 Power-on/reset state ........................................... 170
  5.5 GTZC interrupts ....................................................... 171
  5.6 GTZC1 TZSC registers ................................................ 172
    5.6.1 GTZC1 TZSC control register (GTZC1_TZSC_CR) ............... 172
    5.6.2 GTZC1 TZSC secure configuration register 1
          (GTZC1_TZSC_SECCFGR1) ..................................... 172
    5.6.3 GTZC1 TZSC secure configuration register 2
          (GTZC1_TZSC_SECCFGR2) ..................................... 175
    5.6.4 GTZC1 TZSC secure configuration register 3
          (GTZC1_TZSC_SECCFGR3) ..................................... 177
    5.6.5 GTZC1 TZSC privilege configuration register 1
          (GTZC1_TZSC_PRIVCFGR1) .................................... 179
    5.6.6 GTZC1 TZSC privilege configuration register 2
          (GTZC1_TZSC_PRIVCFGR2) .................................... 181
    5.6.7 GTZC1 TZSC privilege configuration register 3
          (GTZC1_TZSC_PRIVCFGR3) .................................... 184
    5.6.8 GTZC1 TZSC memory x subregion z watermark configuration
          register (GTZC1_TZSC_MPCWMxzCFGR) (z = A to B) ........... 186
    5.6.9 GTZC1 TZSC memory x subregion A watermark register
          (GTZC1_TZSC_MPCWMxAR) ...................................... 187
    5.6.10 GTZC1 TZSC memory x subregion B watermark register
          (GTZC1_TZSC_MPCWMxBR) ...................................... 188
    5.6.11 GTZC1 TZSC register map ..................................... 189
  5.7 GTZC1 TZIC registers ................................................ 191
    5.7.1 GTZC1 TZIC interrupt enable register 1
          (GTZC1_TZIC_IER1) ........................................... 191
    5.7.2 GTZC1 TZIC interrupt enable register 2
          (GTZC1_TZIC_IER2) ........................................... 193
    5.7.3 GTZC1 TZIC interrupt enable register 3
          (GTZC1_TZIC_IER3) ........................................... 195
    5.7.4 GTZC1 TZIC interrupt enable register 4
          (GTZC1_TZIC_IER4) ........................................... 197
    5.7.5 GTZC1 TZIC status register 1
          (GTZC1_TZIC_SR1) ............................................. 199
5.7.6 GTZC1 TZIC status register 2 (GTZC1_TZIC_SR2) ........................................... 202
5.7.7 GTZC1 TZIC status register 3 (GTZC1_TZIC_SR3) ........................................... 204
5.7.8 GTZC1 TZIC status register 4 (GTZC1_TZIC_SR4) ........................................... 206
5.7.9 GTZC1 TZIC flag clear register 1 (GTZC1_TZIC_FCR1) ................................... 208
5.7.10 GTZC1 TZIC flag clear register 2 (GTZC1_TZIC_FCR2) ................................. 210
5.7.11 GTZC1 TZIC flag clear register 3 (GTZC1_TZIC_FCR3) ................................. 212
5.7.12 GTZC1 TZIC flag clear register 4 (GTZC1_TZIC_FCR4) ................................. 214
5.7.13 GTZC1 TZIC register map ............................................................................ 217

5.8 GTZC1 MPCBBz registers (z = 1 to 3) ................................................................. 219

5.8.1 GTZC1 SRAMz MPCBB control register (GTZC1_MPCBBz_CR) (z = 1 to 3) .... 219
5.8.2 GTZC1 SRAMz MPCBB configuration lock register 1 (GTZC1_MPCBBz_CFGLOCK1) (z = 1 to 3) ................................................................. 220
5.8.3 GTZC1 SRAMz MPCBB security configuration for super-block x register (GTZC1_MPCBBz_SECCFGRx) (z = 1 to 3) ................................. 220
5.8.4 GTZC1 SRAMz MPCBB privileged configuration for super-block x register (GTZC1_MPCBBz_PRIVCFGRx) (z = 1 to 3) ................................. 221
5.8.5 GTZC1 MPCBBz register map (z = 1 to 3) ....................................................... 221

6 RAMs configuration controller (RAMCFG) ......................................................... 222

6.1 Introduction ..................................................................................................... 222
6.2 RAMCFG main features ............................................................................... 222
6.3 RAMCFG functional description .................................................................. 222
6.3.1 Internal SRAMs features ............................................................................. 222
6.3.2 Error code correction (SRAM2, SRAM3, BKPSRAM) ............................... 223
6.3.3 Write protection (SRAM2) .......................................................................... 225
6.3.4 Software erase .......................................................................................... 225
6.4 RAMCFG low-power modes .......................................................................... 225
6.5 RAMCFG interrupts .................................................................................... 225
6.6 RAMCFG registers ...................................................................................... 226
6.6.1 RAMCFG memory x control register (RAMCFG_MxCR) ...................... 226
6.6.2 RAMCFG memory x interrupt enable register (RAMCFG_MxEIER) .... 227
6.6.3 RAMCFG memory interrupt status register (RAMCFG_MxISR) .......... 227
6.6.4 RAMCFG memory x ECC single error address register (RAMCFG_MxSEAR) ................................................................. 228
6.6.5 RAMCFG memory x ECC double error address register (RAMCFG_MxDEAR) ................................................................. 228
6.6.6 RAMCFG memory x interrupt clear register x (RAMCFG_MxCICR) ......... 229
6.6.7 RAMCFG memory 2 write protection register 1 (RAMCFG_M2WPR1) ........................................... 229
6.6.8 RAMCFG memory 2 write protection register 2 (RAMCFG_M2WPR2) ........................................... 230
6.6.9 RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR) .................................................. 230
6.6.10 RAMCFG memory x erase key register (RAMCFG_MxERKEYR) ............................................... 231
6.6.11 RAMCFG register map ...................................................... 231

7 Embedded flash memory (FLASH) ........................................ 234
7.1 Introduction ............................................................. 234
7.2 FLASH main features ................................................ 234
7.3 FLASH functional description ........................................ 235
  7.3.1 FLASH block diagram ............................................. 235
  7.3.2 FLASH signals ..................................................... 236
  7.3.3 Flash memory architecture and usage ......................... 237
  7.3.4 FLASH read operations ........................................... 239
  7.3.5 FLASH program operations ..................................... 241
  7.3.6 FLASH erase operations ......................................... 244
  7.3.7 FLASH parallel operations ...................................... 247
  7.3.8 FLASH error protections .......................................... 248
  7.3.9 OTP and RO memory access ..................................... 248
  7.3.10 Flash high-cycle data ............................................ 250
  7.3.11 Flash bank swapping ........................................... 251
  7.3.12 FLASH reset and clocks ......................................... 253
7.4 FLASH option bytes .................................................. 255
  7.4.1 About option bytes ............................................... 255
  7.4.2 Option-byte loading ............................................... 255
  7.4.3 Option-byte modification ....................................... 255
  7.4.4 Description of user and system option bytes ............... 258
  7.4.5 Description of data protection option bytes ............... 259
  7.4.6 Description of boot address option bytes ................. 260
  7.4.7 Specific rules for modifying option bytes ................. 260
7.5 Option-byte key management .......................................... 262
  7.5.1 OBK loading ...................................................... 262
  7.5.2 OBK access per HDPL level ..................................... 263
  7.5.3 OBK programming sequence ..................................... 263
  7.5.4 OBK programming finite state machine ....................... 265
| 7.5.5 | OBK swap sector ........................................... 266 |
| 7.5.6 | OBK alternate sector erase ................................. 269 |
| 7.6 | FLASH security and protections .............................. 269 |
| 7.6.1 | TrustZone security protection .............................. 270 |
| 7.6.2 | Hide protection (HDP) ........................................ 272 |
| 7.6.3 | Block-based secure flash memory area protection .......... 274 |
| 7.6.4 | Block-based privileged flash memory area protection ..... 275 |
| 7.6.5 | Flash memory register privileged and unprivileged modes ...... 276 |
| 7.6.6 | Flash memory banks attributes in case of bank swap ........ 277 |
| 7.6.7 | Flash memory configuration protection ...................... 278 |
| 7.6.8 | Write protection ................................................ 279 |
| 7.6.9 | Flash high-cycle data protections .......................... 279 |
| 7.6.10 | Life cycle management ....................................... 281 |
| 7.6.11 | Product state transitions ..................................... 282 |
| 7.6.12 | OBK protection ................................................. 284 |
| 7.6.13 | One-time-programmable and read-only memory protections .... 285 |
| 7.7 | System memory .................................................. 286 |
| 7.7.1 | Introduction .................................................... 286 |
| 7.7.2 | RSS user functions ............................................. 286 |
| 7.8 | FLASH low-power modes ....................................... 292 |
| 7.9 | FLASH error management ...................................... 293 |
| 7.9.1 | Introduction .................................................... 293 |
| 7.9.2 | Non-secure write protection error (WRPERR) ................ 294 |
| 7.9.3 | Secure write protection error (WRPERR) .................... 294 |
| 7.9.4 | Non secure programming sequence error (PGSERR) .......... 295 |
| 7.9.5 | Secure programming sequence error (PGSERR) ............... 297 |
| 7.9.6 | Non-secure strobe error (STRBERR) ......................... 298 |
| 7.9.7 | Secure strobe error (STRBERR) .............................. 298 |
| 7.9.8 | Non-secure inconsistency error (INCERR) ................. 298 |
| 7.9.9 | Secure inconsistency error (INCERR) ....................... 299 |
| 7.9.10 | Error correction code error (ECCC, ECCD) ................. 299 |
| 7.9.11 | Illegal access (ILAFM/ILAP) .................................. 301 |
| 7.9.12 | Option-byte change error (OPTCHANGEERR) ............... 301 |
| 7.9.13 | Miscellaneous HardFault errors ............................. 302 |
| 7.9.14 | OBK error cases (OBKERR, OBKWERR) ....................... 302 |
| 7.10 | FLASH interrupts ............................................... 303 |
### 7.11 FLASH registers

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.11.1 FLASH access control register (FLASH_ACR)</td>
<td>305</td>
</tr>
<tr>
<td>7.11.2 FLASH non-secure key register (FLASH_NSKEYR)</td>
<td>306</td>
</tr>
<tr>
<td>7.11.3 FLASH secure key register (FLASH_SECKEYR)</td>
<td>306</td>
</tr>
<tr>
<td>7.11.4 FLASH option key register (FLASH_OPTKEYR)</td>
<td>307</td>
</tr>
<tr>
<td>7.11.5 FLASH non-secure OBK key register (FLASH_NSOBKKEYR)</td>
<td>307</td>
</tr>
<tr>
<td>7.11.6 FLASH secure OBK key register (FLASH_SECOBKKEYR)</td>
<td>308</td>
</tr>
<tr>
<td>7.11.7 FLASH operation status register (FLASH_OPSR)</td>
<td>308</td>
</tr>
<tr>
<td>7.11.8 FLASH option control register (FLASH_OPTCR)</td>
<td>309</td>
</tr>
<tr>
<td>7.11.9 FLASH non-secure status register (FLASH_NSSR)</td>
<td>310</td>
</tr>
<tr>
<td>7.11.10 FLASH secure status register (FLASH_SECXR)</td>
<td>310</td>
</tr>
<tr>
<td>7.11.11 FLASH non-secure control register (FLASH_NSCCR)</td>
<td>312</td>
</tr>
<tr>
<td>7.11.12 FLASH secure control register (FLASH_SECXR)</td>
<td>317</td>
</tr>
<tr>
<td>7.11.13 FLASH non-secure clear control register (FLASH_NSCCR)</td>
<td>320</td>
</tr>
<tr>
<td>7.11.14 FLASH secure clear control register (FLASH_SECSCCR)</td>
<td>321</td>
</tr>
<tr>
<td>7.11.15 FLASH privilege configuration register (FLASH_PRIVCFGR)</td>
<td>322</td>
</tr>
<tr>
<td>7.11.16 FLASH non-secure OBK configuration register (FLASH_NSOBKCFGR)</td>
<td>322</td>
</tr>
<tr>
<td>7.11.17 FLASH secure OBK configuration register (FLASH_SECOBKCFGR)</td>
<td>324</td>
</tr>
<tr>
<td>7.11.18 FLASH HDP extension register (FLASH_HDPEXTR)</td>
<td>325</td>
</tr>
<tr>
<td>7.11.19 FLASH option status register (FLASH_OPTSR_CUR)</td>
<td>325</td>
</tr>
<tr>
<td>7.11.20 FLASH option status register (FLASH_OPTSR_PRG)</td>
<td>327</td>
</tr>
<tr>
<td>7.11.21 FLASH non-secure EPOCH register (FLASH_NSEPCHR_CUR)</td>
<td>329</td>
</tr>
<tr>
<td>7.11.22 FLASH secure EPOCH register (FLASH_SECSEPCHR_CUR)</td>
<td>329</td>
</tr>
<tr>
<td>7.11.23 FLASH option status register 2 (FLASH_OPTSR2_CUR)</td>
<td>330</td>
</tr>
<tr>
<td>7.11.24 FLASH option status register 2 (FLASH_OPTSR2_PRG)</td>
<td>331</td>
</tr>
<tr>
<td>7.11.25 FLASH non-secure boot register (FLASH_NSBOOTR_CUR)</td>
<td>332</td>
</tr>
<tr>
<td>7.11.26 FLASH non-secure boot register (FLASH_NSBOOTR_PRG)</td>
<td>332</td>
</tr>
<tr>
<td>7.11.27 FLASH secure boot register (FLASH_SECBOOTR_CUR)</td>
<td>333</td>
</tr>
<tr>
<td>7.11.28 FLASH secure boot register (FLASH_BOOTR_PRG)</td>
<td>334</td>
</tr>
<tr>
<td>7.11.29 FLASH non-secure OTP block lock (FLASH_OTPBLR_CUR)</td>
<td>334</td>
</tr>
<tr>
<td>7.11.30 FLASH non-secure OTP block lock (FLASH_OTPBLR_PRG)</td>
<td>335</td>
</tr>
<tr>
<td>7.11.31 FLASH secure block based register for Bank 1 (FLASH_SECBB1Rx)</td>
<td>335</td>
</tr>
<tr>
<td>7.11.32 FLASH privilege block based register for Bank 1 (FLASH_PRIVBB1Rx)</td>
<td>335</td>
</tr>
<tr>
<td>7.11.33 FLASH security watermark for Bank 1 (FLASH_SECWM1R_CUR)</td>
<td>336</td>
</tr>
<tr>
<td>7.11.34 FLASH security watermark for Bank 1 (FLASH_SECWM1R_PRG)</td>
<td>337</td>
</tr>
</tbody>
</table>
7.11.35 FLASH write sector group protection for Bank 1 (FLASH_WRP1R_CUR) ........................................ 337
7.11.36 FLASH write sector group protection for Bank 1 (FLASH_WRP1R_PRG) ....................................... 338
7.11.37 FLASH data sector configuration Bank 1 (FLASH_EDATA1R_CUR) ................................. 338
7.11.38 FLASH data sector configuration Bank 1 (FLASH_EDATA1R_PRG) ........................................ 339
7.11.39 FLASH HDP Bank 1 configuration (FLASH_HDP1R_CUR) ..................................................... 340
7.11.40 FLASH HDP Bank 1 configuration (FLASH_HDP1R_PRG) ..................................................... 340
7.11.41 FLASH ECC correction register (FLASH_ECCCORR) ......................................................... 341
7.11.42 FLASH ECC detection register (FLASH_ECCDETR) ............................................................ 342
7.11.43 FLASH ECC data (FLASH_ECCDR) .................................................................................. 343
7.11.44 FLASH secure block-based register for Bank 2 (FLASH_SECBB2Rx) .................................. 343
7.11.45 FLASH privilege block-based register for Bank 2 (FLASH_PRIVBB2Rx) ............................... 344
7.11.46 FLASH security watermark for Bank 2 (FLASH_SECWM2R_CUR) .................................. 344
7.11.47 FLASH security watermark for Bank 2 (FLASH_SECWM2R_PRG) .................................. 345
7.11.48 FLASH write sector group protection for Bank 2 (FLASH_WRP2R_CUR) .......................... 345
7.11.49 FLASH write sector group protection for Bank 2 (FLASH_WRP2R_PRG) .......................... 346
7.11.50 FLASH data sectors configuration Bank 2 (FLASH_EDATA2R_CUR) .......................... 346
7.11.51 FLASH data sector configuration Bank 2 (FLASH_EDATA2R_PRG) ...................................... 347
7.11.52 FLASH HDP Bank 2 configuration (FLASH_HDP2R_CUR) ................................................. 348
7.11.53 FLASH HDP Bank 2 configuration (FLASH_HDP2R_PRG) ................................................. 348
7.12 FLASH register map and reset values ................................................................. 349

8 Instruction cache (ICACHE) ......................................................... 355
8.1 ICACHE introduction ................................................................. 355
8.2 ICACHE main features ............................................................... 355
8.3 ICACHE implementation ............................................................ 356
8.4 ICACHE functional description .................................................... 356
8.4.1 ICACHE block diagram .......................................................... 357
8.4.2 ICACHE reset and clocks ......................................................... 357
8.4.3 ICACHE TAG memory ............................................................. 358
8.4.4 Direct-mapped ICACHE (1-way cache) ........................................ 359
8.4.5 ICACHE enable ................................................................. 360
8.4.6 Cacheable and non-cacheable traffic ........................................ 360
8.4.7 Address remapping ............................................. 361
8.4.8 Cacheable accesses ........................................ 363
8.4.9 Dual-master cache ....................................... 364
8.4.10 ICACHE security ......................................... 364
8.4.11 ICACHE maintenance .................................. 364
8.4.12 ICACHE performance monitoring ..................... 365
8.4.13 ICACHE boot ............................................. 365
8.5 ICACHE low-power modes .................................. 365
8.6 ICACHE error management and interrupts ............... 366
8.7 ICACHE registers ............................................. 366
8.7.1 ICACHE control register (ICACHE_CR) .................. 366
8.7.2 ICACHE status register (ICACHE_SR) .................... 367
8.7.3 ICACHE interrupt enable register (ICACHE_IER) ....... 368
8.7.4 ICACHE flag clear register (ICACHE_FCR) ............. 368
8.7.5 ICACHE hit monitor register (ICACHE_HMONR) ........ 369
8.7.6 ICACHE miss monitor register (ICACHE_MMONR) ...... 369
8.7.7 ICACHE region x configuration register (ICACHE_CRRx) 369
8.7.8 ICACHE register map .................................. 370

9 Data cache (DCACHE) ............................................. 372
9.1 DCACHE introduction ........................................ 372
9.2 DCACHE main features ...................................... 372
9.3 DCACHE implementation .................................... 373
9.4 DCACHE functional description ............................. 373
9.4.1 DCACHE block diagram ................................ 374
9.4.2 DCACHE reset and clocks ............................... 374
9.4.3 DCACHE TAG memory .................................. 375
9.4.4 DCACHE enable .......................................... 377
9.4.5 Cacheable and non-cacheable traffic ................. 377
9.4.6 Cacheable accesses .................................... 378
9.4.7 DCACHE security ....................................... 380
9.4.8 DCACHE maintenance .................................. 380
9.4.9 DCACHE performance monitoring ..................... 382
9.4.10 DCACHE boot .......................................... 382
9.5 DCACHE low-power modes .................................. 382
9.6 DCACHE error management and interrupts ............... 383
9.7 DCACHE registers .................................................. 384
  9.7.1 DCACHE control register (DCACHE_CR) ................. 384
  9.7.2 DCACHE status register (DCACHE_SR) .................. 385
  9.7.3 DCACHE interrupt enable register (DCACHE_IER) ....... 386
  9.7.4 DCACHE flag clear register (DCACHE_FCR) ........... 387
  9.7.5 DCACHE read-hit monitor register (DCACHE_RHMONR) .... 387
  9.7.6 DCACHE read-miss monitor register (DCACHE_RMMONR) .... 388
  9.7.7 DCACHE write-hit monitor register (DCACHE_WHMONR) ... 388
  9.7.8 DCACHE write-miss monitor register (DCACHE_WMMONR) ... 388
  9.7.9 DCACHE command range start address register
        (DCACHE_CMDRSADDRR) ..................................... 389
  9.7.10 DCACHE command range end address register
        (DCACHE_CMDREADDRR) ..................................... 389
  9.7.11 DCACHE register map ..................................... 389

10 Power control (PWR) .............................................. 391
  10.1 Introduction .................................................. 391
  10.2 PWR main features .......................................... 391
  10.3 PWR pins and internal signals ............................. 392
  10.4 PWR power supplies and supply domains ................... 393
    10.4.1 External power supplies .................................. 394
    10.4.2 Internal regulators ...................................... 395
    10.4.3 Power-up and power-down power sequences ............. 397
    10.4.4 Independent analog peripherals supply ................ 397
    10.4.5 Independent I/O supply rail ............................ 398
    10.4.6 Independent USB transceivers supply ................. 398
    10.4.7 Backup domain ........................................... 398
  10.5 PWR system supply voltage regulation .................... 400
    10.5.1 SMPS and LDO embedded regulators ..................... 400
    10.5.2 $V_{\text{CORE}}$ supply versus reset, voltage scaling, and low-power modes .... 400
    10.5.3 Embedded voltage regulator operating modes .......... 400
  10.6 PWR power supply and temperature supervision .......... 401
    10.6.1 Power-on reset (POR)/power-down reset (PDR) ....... 401
    10.6.2 Brownout reset (BOR), ................................... 401
    10.6.3 Programmable voltage detector (PVD) ................. 402
    10.6.4 Analog voltage detector (AVD) .......................... 403
    10.6.5 VDDIO2 voltage monitor (IO2VM) ....................... 404
10.6.6 Backup domain voltage monitoring .......................... 404
10.6.7 Temperature monitoring ...................................... 405

10.7 PWR management .................................................. 406
  10.7.1 Voltage scaling .................................................. 406
  10.7.2 Power management examples ................................. 407

10.8 Power modes .......................................................... 408
  10.8.1 Slowing down system clocks ...................................... 412
  10.8.2 Peripheral clock gating .......................................... 412
  10.8.3 Low-power modes ................................................ 412
  10.8.4 Sleep mode ....................................................... 413
  10.8.5 Stop mode ....................................................... 414
  10.8.6 Standby mode .................................................... 416
  10.8.7 Power modes output pins ........................................ 419

10.9 PWR security and privileged protection ......................... 419
  10.9.1 PWR security protection ....................................... 419
  10.9.2 PWR privileged protection ..................................... 421

10.10 PWR interrupts .................................................... 421

10.11 PWR registers ...................................................... 423
  10.11.1 PWR power mode control register (PWR_PMCR) ............... 423
  10.11.2 PWR status register (PWR_PMSR) .............................. 425
  10.11.3 PWR voltage scaling control register (PWR_VOSCR) ........ 425
  10.11.4 PWR voltage scaling status register (PWR_VOSSR) .......... 426
  10.11.5 PWR Backup domain control register (PWR_BDCR) .......... 427
  10.11.6 PWR Backup domain control register (PWR_DBPCR) .......... 428
  10.11.7 PWR Backup domain status register (PWR_BDSR) .......... 428
  10.11.8 PWR USB Type-C power delivery register (PWR_UCPDR) ..... 429
  10.11.9 PWR supply configuration control register (PWR_SCCR) ... 430
  10.11.10 PWR voltage monitor control register (PWR_VMCR) ....... 430
  10.11.11 PWR USB supply control register (PWR_USBSCR) .......... 431
  10.11.12 PWR voltage monitor status register (PWR_VMSR) ....... 432
  10.11.13 PWR wake-up status clear register (PWR_WUSCR) .......... 433
  10.11.14 PWR wake-up status register (PWR_WUSR) .................. 433
  10.11.15 PWR wake-up configuration register (PWR_WUCR) .......... 434
  10.11.16 PWR I/O retention register (PWR_IORETR) .................. 435
  10.11.17 PWR security configuration register (PWR_SECCFG) ....... 436
  10.11.18 PWR privilege configuration register (PWR_PRIVCFG) ..... 437
11 Reset and clock control (RCC) ................................................. 440

11.1 Introduction ............................................................... 440
11.2 RCC pins and internal signals ......................................... 440
11.3 RCC reset functional description ...................................... 440
  11.3.1 Power reset ......................................................... 440
  11.3.2 System reset ....................................................... 441
  11.3.3 Backup domain reset ............................................. 442
  11.3.4 Reset source identification .................................... 442
11.4 RCC clocks functional description ..................................... 443
  11.4.1 HSE clock ........................................................... 445
  11.4.2 HSI clock ........................................................... 446
  11.4.3 CSI oscillator ...................................................... 447
  11.4.4 HSI48 clock ........................................................ 448
  11.4.5 PLL description .................................................... 448
  11.4.6 LSE clock ........................................................... 452
  11.4.7 LSI clock ........................................................... 453
  11.4.8 System clock (SYSCLK) selection ............................... 454
  11.4.9 Handling clock generators in stop and standby modes ...... 454
  11.4.10 Clock security system (CSS) ................................... 455
  11.4.11 Clock output generation (MCO1/MCO2) ....................... 456
  11.4.12 Kernel clock selection .......................................... 457
  11.4.13 RTC and TAMP clock ............................................ 460
  11.4.14 Timer clock ........................................................ 461
  11.4.15 Watchdog clock .................................................. 461
  11.4.16 Peripherals clock gating and autonomous mode ............ 461
11.5 RCC security and privilege functional description .................. 462
  11.5.1 RCC TrustZone security protection modes ....................... 462
  11.5.2 RCC privilege protection modes ................................ 465
11.6 RCC low-power modes .................................................... 465
11.7 RCC interrupts .......................................................... 467
11.8 RCC registers ............................................................. 469
  11.8.1 RCC clock control register (RCC_CR) ......................... 469
  11.8.2 RCC HSI calibration register (RCC_HSI_CFGR) .............. 472
  11.8.3 RCC clock recovery RC register (RCC_CRRCR) ............... 473
11.8.4 RCC CSI calibration register (RCC_CSI_CFGR) .......................... 473
11.8.5 RCC clock configuration register1 (RCC_CFGR1) ......................... 474
11.8.6 RCC CPU domain clock configuration register 2 (RCC_CFGR2) ...... 476
11.8.7 RCC PLL clock source selection register (RCC_PLL1CFGR) .......... 479
11.8.8 RCC PLL clock source selection register (RCC_PLL2CFGR) .......... 481
11.8.9 RCC PLL clock source selection register (RCC_PLL3CFGR) .......... 482
11.8.10 RCC PLL1 dividers register (RCC_PLL1DIVR) ............................ 484
11.8.11 RCC PLL1 fractional divider register (RCC_PLL1FRACR) ............. 485
11.8.12 RCC PLL1 dividers register (RCC_PLL2DIVR) ............................ 486
11.8.13 RCC PLL2 fractional divider register (RCC_PLL2FRACR) ............. 487
11.8.14 RCC PLL3 dividers register (RCC_PLL3DIVR) ............................ 488
11.8.15 RCC PLL3 fractional divider register (RCC_PLL3FRACR) ............. 489
11.8.16 RCC clock source interrupt enable register (RCC_CIER) .......... 490
11.8.17 RCC clock source interrupt flag register (RCC_CIFR) ............... 491
11.8.18 RCC clock source interrupt clear register (RCC_CICR) ............. 493
11.8.19 RCC AHB1 reset register (RCC_AHB1RSTR) ............................. 494
11.8.20 RCC AHB2 peripheral reset register (RCC_AHB2RSTR) .............. 495
11.8.21 RCC AHB4 peripheral reset register (RCC_AHB4RSTR) ............... 497
11.8.22 RCC APB1 peripheral low reset register (RCC_APB1LRSTR) ........ 498
11.8.23 RCC APB1 peripheral high reset register (RCC_APB1HRSTR) ....... 501
11.8.24 RCC APB2 peripheral reset register (RCC_APB2RSTR) ............... 502
11.8.25 RCC APB3 peripheral reset register (RCC_APB3RSTR) ............... 503
11.8.26 RCC AHB1 peripherals clock register (RCC_AHB1ENR) ............... 505
11.8.27 RCC AHB2 peripheral clock register (RCC_AHB2ENR) ............... 506
11.8.28 RCC AHB4 peripheral clock register (RCC_AHB4ENR) ............... 509
11.8.29 RCC APB1 peripheral clock register (RCC_APB1LENR) ............... 510
11.8.30 RCC APB1 peripheral clock register (RCC_APB1HENR) ............... 512
11.8.31 RCC APB2 peripheral clock register (RCC_APB2ENR) ............... 513
11.8.32 RCC APB3 peripheral clock register (RCC_APB3ENR) ............... 515
11.8.33 RCC AHB1 sleep clock register (RCC_AHB1LPENR) ................... 516
11.8.34 RCC AHB2 sleep clock register (RCC_AHB2LPENR) ................... 518
11.8.35 RCC AHB4 sleep clock register (RCC_AHB4LPENR) .................. 520
11.8.36 RCC APB1 sleep clock register (RCC_APB1LLPENR) .................. 521
11.8.37 RCC APB1 sleep clock register (RCC_APB1HLPENR) .................. 524
11.8.38 RCC APB2 sleep clock register (RCC_APB2LPENR) .................. 525
11.8.39 RCC APB3 sleep clock register (RCC_APB3LPENR) .................. 526
11.8.40 RCC kernel clock configuration register (RCC_CCIPR1) .......... 528
11.8.41 RCC kernel clock configuration register (RCC_CCIPR2) ............... 530
11.8.42 RCC kernel clock configuration register (RCC_CCIPR3) ............... 532
11.8.43 RCC kernel clock configuration register (RCC_CCIPR4) ............... 534
11.8.44 RCC kernel clock configuration register (RCC_CCIPR5) ............... 535
11.8.45 RCC Backup domain control register (RCC_BDCR) ....................... 536
11.8.46 RCC reset status register (RCC_RSR) .................................. 539
11.8.47 RCC secure configuration register (RCC_SECCFGR) ...................... 540
11.8.48 RCC privilege configuration register (RCC_PRIVCFGR) ................. 542
11.9 RCC register map .......................................................... 543

12 Clock recovery system (CRS) ................................................. 549
12.1 Introduction ................................................................. 549
12.2 CRS main features .......................................................... 549
12.3 CRS implementation ........................................................ 549
12.4 CRS functional description .................................................. 550
  12.4.1 CRS block diagram ...................................................... 550
  12.4.2 Synchronization input .................................................. 550
  12.4.3 Frequency error measurement ........................................ 551
  12.4.4 Frequency error evaluation and automatic trimming .................... 552
  12.4.5 CRS initialization and configuration .................................. 553
12.5 CRS low-power modes ....................................................... 554
12.6 CRS interrupts ............................................................... 554
12.7 CRS registers ................................................................. 555
  12.7.1 CRS control register (CRS_CR) ....................................... 555
  12.7.2 CRS configuration register (CRS_CFGFR) ............................... 556
  12.7.3 CRS interrupt and status register (CRS_ISR) ......................... 557
  12.7.4 CRS interrupt flag clear register (CRS_ICR) ....................... 559
  12.7.5 CRS register map ...................................................... 559

13 General-purpose I/Os (GPIO) ................................................. 561
13.1 Introduction ................................................................. 561
13.2 GPIO main features ........................................................ 561
13.3 GPIO functional description .................................................. 561
  13.3.1 General-purpose I/O (GPIO) ........................................... 563
  13.3.2 I/O pin alternate function multiplexer and mapping ................... 564
  13.3.3 I/O port control registers ............................................. 564
13.3.4 I/O port data registers ........................................... 565
13.3.5 I/O data bitwise handling ........................................ 565
13.3.6 GPIO locking mechanism ....................................... 565
13.3.7 I/O alternate function input/output ............................. 565
13.3.8 External interrupt/wakeup lines ................................. 566
13.3.9 Input configuration ............................................. 566
13.3.10 Output configuration ........................................... 567
13.3.11 Alternate function configuration ............................. 567
13.3.12 Analog configuration .......................................... 568
13.3.13 Using the HSE or LSE oscillator pins as GPIOs .............. 568
13.3.14 Using the GPIO pins in the RTC supply domain ............... 569
13.3.15 I/Os state retention during standby mode ..................... 569
13.3.16 TrustZone security ........................................... 569
13.3.17 Privileged and unprivileged modes .......................... 570
13.3.18 High-speed low-voltage mode (HSLV) ......................... 570
13.3.19 I/O compensation cell ....................................... 571

13.4 GPIO registers .................................................. 571
13.4.1 GPIO port mode register (GPIOx_MODER) (x = A to I) ...... 571
13.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to I) .... 571
13.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to I) .... 572
13.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to I) .... 572
13.4.5 GPIO port input data register (GPIOx_IDR) (x = A to I) ........ 573
13.4.6 GPIO port output data register (GPIOx_ODR) (x = A to I) ....... 573
13.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to I) .... 574
13.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to I) .... 574
13.4.9 GPIO alternate function low register (GPIOx_AFRDL) (x = A to I) .... 575
13.4.10 GPIO alternate function high register (GPIOx_AFRL) (x = A to H) .... 576
13.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to I) ........ 577
13.4.12 GPIO high-speed low-voltage register (GPIOx_HSLVR) (x = A to I) .... 578
13.4.13 GPIO secure configuration register (GPIOx_SECCFGR) (x = A to I) .... 578
13.4.14 GPIO register map ......................................... 579

14 System configuration, boot, and security (SBS) ............... 581
14.1 SBS introduction ............................................. 581
14.2 SBS main features ............................................. 581
14.3 SBS functional description ................................. 582
14.3.1 SBS block diagram ....................................... 582
14.3.2 SBS signals ................................................. 583
14.3.3 SBS reset and clocks .................................... 583
14.3.4 SBS system configuration ................................. 584
14.3.5 SBS boot control ......................................... 585
14.3.6 SBS debug control ....................................... 588
14.3.7 SBS hardware secure storage control ................. 590
14.4 SBS interrupts ............................................... 592
14.5 SBS registers ................................................ 592
  14.5.1 SBS temporal isolation control register (SBS_HDPLCR) .... 592
  14.5.2 SBS temporal isolation status register (SBS_HDPLSR) .... 593
  14.5.3 SBS next HDPL control register (SBS_NEXTHDPLCR) .... 593
  14.5.4 SBS debug control register (SBS_DBGCR) ............... 594
  14.5.5 SBS debug lock register (SBS_DBGLOCKR) ............. 594
  14.5.6 SBS RSS command register (SBS_RSSCMRD) .............. 595
  14.5.7 SBS EPOCH selection control register (SBS_EPOCHSELCR) . 595
  14.5.8 SBS security mode configuration control register (SBS_SECCFGR) . 596
  14.5.9 SBS product mode and configuration register (SBS_PMCR) ...... 597
  14.5.10 SBS CPU interrupt mask register (SBS_FPUIMR) ......... 598
  14.5.11 SBS memory erase status register (SBS_MESR) ........... 599
  14.5.12 SBS compensation cell for I/Os control and status register (SBS_CCCSR) .......... 599
  14.5.13 SBS compensation cell for I/Os value register (SBS_CCVALR) .... 600
  14.5.14 SBS compensation cell for I/Os software code register (SBS_CCSWCR) .......... 601
  14.5.15 SBS Class B register (SBS_CFGR2) .................... 602
  14.5.16 SBS CPU non-secure lock register (SBS_CNSLCKR) ....... 603
  14.5.17 SBS CPU secure lock register (SBS_CSLCKR) ............ 604
  14.5.18 SBS flift ECC NMI mask register (SBS_ECCNMIR) ....... 605
  14.5.19 SBS register map ..................................... 606

15 Peripherals interconnect matrix ................................ 608
  15.1 Interconnect matrix introduction .......................... 608
  15.2 Connection summary ....................................... 609
  15.3 Interconnection details .................................... 611
    15.3.1 Master to slave interconnection for timers ............ 611
    15.3.2 Triggers to ADCs ..................................... 611
    15.3.3 ADC analog watchdogs as triggers to timers .......... 612
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.3.4</td>
<td>Triggers to DAC</td>
<td>613</td>
</tr>
<tr>
<td>15.3.5</td>
<td>Clock sources to timers</td>
<td>613</td>
</tr>
<tr>
<td>15.3.6</td>
<td>Triggers to low-power timers</td>
<td>614</td>
</tr>
<tr>
<td>15.3.7</td>
<td>RTC wake-up as inputs to timers</td>
<td>614</td>
</tr>
<tr>
<td>15.3.8</td>
<td>System errors as break signals to timers</td>
<td>615</td>
</tr>
<tr>
<td>15.3.9</td>
<td>Triggers for communication peripherals</td>
<td>615</td>
</tr>
<tr>
<td>15.3.10</td>
<td>Triggers to GPDMA1/2</td>
<td>616</td>
</tr>
<tr>
<td>15.3.11</td>
<td>Internal analog signals to analog peripherals</td>
<td>616</td>
</tr>
<tr>
<td>15.3.12</td>
<td>Clock source for the DAC sample and hold mode</td>
<td>617</td>
</tr>
<tr>
<td>15.3.13</td>
<td>Internal tamper sources</td>
<td>617</td>
</tr>
<tr>
<td>15.3.14</td>
<td>Output from tamper to RTC</td>
<td>617</td>
</tr>
<tr>
<td>15.3.15</td>
<td>Encryption keys to AES/SAES</td>
<td>618</td>
</tr>
</tbody>
</table>

16. General purpose direct memory access controller (GPDMA) | 619

16.1 GPDMA introduction | 619
16.2 GPDMA main features | 620
16.3 GPDMA implementation | 620
  16.3.1 GPDMA instances | 620
  16.3.2 GPDMA channels | 620
  16.3.3 GPDMA autonomous mode in low-power modes | 621
  16.3.4 GPDMA requests | 621
  16.3.5 GPDMA block requests | 625
  16.3.6 GPDMA channels with peripheral early termination | 626
  16.3.7 GPDMA triggers | 626
16.4 GPDMA functional description | 628
  16.4.1 GPDMA block diagram | 628
  16.4.2 GPDMA channel state and direct programming without any linked-list | 628
  16.4.3 GPDMA channel suspend and resume | 629
  16.4.4 GPDMA channel abort and restart | 630
  16.4.5 GPDMA linked-list data structure | 631
  16.4.6 Linked-list item transfer execution | 635
  16.4.7 GPDMA channel state and linked-list programming in run-to-completion mode | 636
  16.4.8 GPDMA channel state and linked-list programming in link step mode | 639
  16.4.9 GPDMA channel state and linked-list programming | 646
  16.4.10 GPDMA FIFO-based transfers | 648
  16.4.11 GPDMA transfer request and arbitration | 655
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.4.12</td>
<td>GPDMA triggered transfer</td>
<td>659</td>
</tr>
<tr>
<td>16.4.13</td>
<td>GPDMA circular buffering with linked-list programming</td>
<td>660</td>
</tr>
<tr>
<td>16.4.14</td>
<td>GPDMA transfer in peripheral flow-control mode</td>
<td>662</td>
</tr>
<tr>
<td>16.4.15</td>
<td>GPDMA secure/nonsecure channel</td>
<td>663</td>
</tr>
<tr>
<td>16.4.16</td>
<td>GPDMA privileged/unprivileged channel</td>
<td>664</td>
</tr>
<tr>
<td>16.4.17</td>
<td>GPDMA error management</td>
<td>664</td>
</tr>
<tr>
<td>16.4.18</td>
<td>GPDMA autonomous mode</td>
<td>666</td>
</tr>
<tr>
<td>16.5</td>
<td>GPDMA in debug mode</td>
<td>667</td>
</tr>
<tr>
<td>16.6</td>
<td>GPDMA in low-power modes</td>
<td>667</td>
</tr>
<tr>
<td>16.7</td>
<td>GPDMA interrupts</td>
<td>668</td>
</tr>
<tr>
<td>16.8</td>
<td>GPDMA registers</td>
<td>669</td>
</tr>
<tr>
<td>16.8.1</td>
<td>GPDMA secure configuration register (GPDMA_SECCFGR)</td>
<td>669</td>
</tr>
<tr>
<td>16.8.2</td>
<td>GPDMA privileged configuration register (GPDMA_PRIVCFGR)</td>
<td>670</td>
</tr>
<tr>
<td>16.8.3</td>
<td>GPDMA configuration lock register (GPDMA_RCFGLOCKR)</td>
<td>670</td>
</tr>
<tr>
<td>16.8.4</td>
<td>GPDMA nonsecure masked interrupt status register (GPDMA_MISR)</td>
<td>671</td>
</tr>
<tr>
<td>16.8.5</td>
<td>GPDMA secure masked interrupt status register (GPDMA_SMISR)</td>
<td>672</td>
</tr>
<tr>
<td>16.8.6</td>
<td>GPDMA channel x linked-list base address register (GPDMA_CxLBAR)</td>
<td>673</td>
</tr>
<tr>
<td>16.8.7</td>
<td>GPDMA channel x flag clear register (GPDMA_CxFCR)</td>
<td>673</td>
</tr>
<tr>
<td>16.8.8</td>
<td>GPDMA channel x status register (GPDMA_CxSR)</td>
<td>674</td>
</tr>
<tr>
<td>16.8.9</td>
<td>GPDMA channel x control register (GPDMA_CxCR)</td>
<td>676</td>
</tr>
<tr>
<td>16.8.10</td>
<td>GPDMA channel x transfer register 1 (GPDMA_CxTR1)</td>
<td>678</td>
</tr>
<tr>
<td>16.8.11</td>
<td>GPDMA channel x transfer register 2 (GPDMA_CxTR2)</td>
<td>682</td>
</tr>
<tr>
<td>16.8.12</td>
<td>GPDMA channel x block register 1 (GPDMA_CxBR1)</td>
<td>686</td>
</tr>
<tr>
<td>16.8.13</td>
<td>GPDMA channel x alternate block register 1 (GPDMA_CxBR1)</td>
<td>687</td>
</tr>
<tr>
<td>16.8.14</td>
<td>GPDMA channel x source address register (GPDMA_CxSAR)</td>
<td>690</td>
</tr>
<tr>
<td>16.8.15</td>
<td>GPDMA channel x destination address register (GPDMA_CxDAR)</td>
<td>692</td>
</tr>
<tr>
<td>16.8.16</td>
<td>GPDMA channel x transfer register 3 (GPDMA_CxTR3)</td>
<td>693</td>
</tr>
<tr>
<td>16.8.17</td>
<td>GPDMA channel x block register 2 (GPDMA_CxBR2)</td>
<td>694</td>
</tr>
<tr>
<td>16.8.18</td>
<td>GPDMA channel x linked-list address register (GPDMA_CxLLR)</td>
<td>695</td>
</tr>
<tr>
<td>16.8.19</td>
<td>GPDMA channel x alternate linked-list address register (GPDMA_CxLLR)</td>
<td>697</td>
</tr>
<tr>
<td>16.8.20</td>
<td>GPDMA register map</td>
<td>698</td>
</tr>
</tbody>
</table>

17 **Nested vectored interrupt controller (NVIC)** ........................................... 701

17.1 NVIC main features .................................................................................. 701
18 Extended interrupts and event controller (EXTI) .................. 707

18.1 EXTI main features .............................................. 707
18.2 EXTI block diagram .............................................. 707
  18.2.1 EXTI connections between peripherals and CPU .......... 709
  18.2.2 EXTI interrupt/event mapping ............................. 709
18.3 EXTI functional description .................................... 711
  18.3.1 EXTI configurable event input wakeup .................. 711
  18.3.2 EXTI direct event input wake-up .......................... 712
  18.3.3 EXTI mux selection ......................................... 712
18.4 EXTI functional behavior ....................................... 713
18.5 EXTI event protection .......................................... 714
  18.5.1 EXTI security protection ................................... 714
  18.5.2 EXTI privilege protection .................................. 715
18.6 EXTI registers .................................................. 716
  18.6.1 EXTI rising trigger selection register (EXTI_RTSR1) .... 716
  18.6.2 EXTI falling trigger selection register (EXTI_FTSR1) .... 716
  18.6.3 EXTI software interrupt event register (EXTI_SWIER1) ... 717
  18.6.4 EXTI rising edge pending register (EXTI_RPR1) ........ 718
  18.6.5 EXTI falling edge pending register (EXTI_FPR1) ........ 719
  18.6.6 EXTI security configuration register (EXTI_SECCFGGR1) ... 719
  18.6.7 EXTI privilege configuration register (EXTI_PRIVCFGR1) ... 720
  18.6.8 EXTI rising trigger selection register 2 (EXTI_RTSR2) ... 720
  18.6.9 EXTI falling trigger selection register 2 (EXTI_FTSR2) ... 721
  18.6.10 EXTI software interrupt event register 2 (EXTI_SWIER2) ... 723
  18.6.11 EXTI rising edge pending register 2 (EXTI_RPR2) ........ 724
  18.6.12 EXTI falling edge pending register 2 (EXTI_FPR2) ........ 725
  18.6.13 EXTI security configuration register 2 (EXTI_SECCFGGR2) ... 727
  18.6.14 EXTI privilege configuration register 2 (EXTI_PRIVCFGR2) ... 727
  18.6.15 EXTI external interrupt selection register (EXTI_EXTICR1) ... 728
  18.6.16 EXTI external interrupt selection register (EXTI_EXTICR2) ... 730
  18.6.17 EXTI external interrupt selection register (EXTI_EXTICR3) ... 733
  18.6.18 EXTI external interrupt selection register (EXTI_EXTICR4) ... 735
  18.6.19 EXTI lock register (EXTI_LOCKR) ......................... 738
19 Cyclic redundancy check calculation unit (CRC) ........................................ 743
  19.1 Introduction ................................................................. 743
  19.2 CRC main features ....................................................... 743
  19.3 CRC functional description ............................................ 744
    19.3.1 CRC block diagram .................................................. 744
    19.3.2 CRC internal signals ............................................... 744
    19.3.3 CRC operation ....................................................... 744
  19.4 CRC registers ............................................................ 746
    19.4.1 CRC data register (CRC_DR) ....................................... 746
    19.4.2 CRC independent data register (CRC_IDR) ....................... 746
    19.4.3 CRC control register (CRC_CR) ................................... 747
    19.4.4 CRC initial value (CRC_INIT) .................................... 748
    19.4.5 CRC polynomial (CRC_POL) ........................................ 748
    19.4.6 CRC register map .................................................... 749

20 CORDIC co-processor (CORDIC) ............................................... 750
  20.1 CORDIC introduction ................................................... 750
  20.2 CORDIC main features .................................................. 750
  20.3 CORDIC functional description ....................................... 750
    20.3.1 General description ................................................ 750
    20.3.2 CORDIC functions ................................................... 750
    20.3.3 Fixed point representation ....................................... 757
    20.3.4 Scaling factor ....................................................... 757
    20.3.5 Precision ............................................................ 758
    20.3.6 Zero-overhead mode ............................................... 761
    20.3.7 Poling mode ........................................................ 762
    20.3.8 Interrupt mode ..................................................... 763
    20.3.9 DMA mode ........................................................... 763
  20.4 CORDIC registers ......................................................... 764
    20.4.1 CORDIC control/status register (CORDIC_CSR) ............... 764
20.4.2 CORDIC argument register (CORDIC_WDATA) ..................... 766
20.4.3 CORDIC result register (CORDIC_RDATA) ....................... 767
20.4.4 CORDIC register map ............................................. 767

21 Filter math accelerator (FMAC) ....................................... 768
21.1 FMAC introduction .................................................... 768
21.2 FMAC main features ................................................ 768
21.3 FMAC functional description ....................................... 769
   21.3.1 General description ............................................. 769
   21.3.2 Local memory and buffers .................................. 770
   21.3.3 Input buffers ................................................ 770
   21.3.4 Output buffer ................................................ 773
   21.3.5 Initialization functions ...................................... 775
   21.3.6 Filter functions .............................................. 776
   21.3.7 Fixed point representation .................................. 780
   21.3.8 Implementing FIR filters with the FMAC .................. 780
   21.3.9 Implementing IIR filters with the FMAC .................. 782
   21.3.10 Examples of filter initialization ......................... 784
   21.3.11 Examples of filter operation ............................... 785
   21.3.12 Filter design tips ........................................... 787
21.4 FMAC registers ..................................................... 788
   21.4.1 FMAC X1 buffer configuration register (FMAC_X1BUFCFG) ... 788
   21.4.2 FMAC X2 buffer configuration register (FMAC_X2BUFCFG) ... 788
   21.4.3 FMAC Y buffer configuration register (FMAC_YBUFCFG) ... 789
   21.4.4 FMAC parameter register (FMAC_PARAM) .................... 790
   21.4.5 FMAC control register (FMAC_CR) .......................... 791
   21.4.6 FMAC status register (FMAC_SR) ............................ 792
   21.4.7 FMAC write data register (FMAC_WDATA) .................. 793
   21.4.8 FMAC read data register (FMAC_RDATA) .................... 794
   21.4.9 FMAC register map ........................................... 794

22 Flexible static memory controller (FSMC) .......................... 796
22.1 Introduction ....................................................... 796
22.2 FMC main features ................................................ 796
22.3 FMC block diagram ................................................ 797
22.4 AHB interface ..................................................... 798
22.4.1 Supported memories and transactions ............................... 799
22.5 External device address mapping .................................. 800
  22.5.1 NOR/PSRAM address mapping ................................. 801
  22.5.2 NAND flash memory address mapping ............................ 801
  22.5.3 SDRAM address mapping .................................... 802
22.6 NOR flash/PSRAM controller ....................................... 804
  22.6.1 External memory interface signals ............................ 806
  22.6.2 Supported memories and transactions ......................... 808
  22.6.3 General timing rules ....................................... 809
  22.6.4 NOR flash/PSRAM controller asynchronous transactions .... 809
  22.6.5 Synchronous transactions ................................... 827
  22.6.6 NOR/PSRAM controller registers ............................. 834
22.7 NAND flash controller ............................................. 842
  22.7.1 External memory interface signals ............................ 842
  22.7.2 NAND flash supported memories and transactions ............ 843
  22.7.3 Timing diagrams for NAND flash memory ....................... 844
  22.7.4 NAND flash operations ..................................... 845
  22.7.5 NAND flash prewait functionality ............................. 845
  22.7.6 Computation of the error correction code (ECC)
in NAND flash memory .............................................. 846
  22.7.7 NAND flash controller registers ............................. 847
22.8 SDRAM controller .................................................. 853
  22.8.1 SDRAM controller main features .............................. 853
  22.8.2 SDRAM External memory interface signals ..................... 853
  22.8.3 SDRAM controller functional description ..................... 854
  22.8.4 Low-power modes ........................................... 860
  22.8.5 SDRAM controller registers ................................ 863
  22.8.6 FMC register map ........................................ 869

23 Octo-SPI interface (OCTOSPI) ........................................ 872
  23.1 Introduction .................................................. 872
  23.2 OCTOSPI main features ........................................ 872
  23.3 OCTOSPI implementation ...................................... 873
  23.4 OCTOSPI functional description ................................ 874
    23.4.1 OCTOSPI block diagram ................................... 874
    23.4.2 OCTOSPI pins and internal signals ......................... 875
23.4.3 OCTOSPI interface to memory modes ........................................ 876
23.4.4 OCTOSPI regular-command protocol ........................................ 876
23.4.5 OCTOSPI regular-command protocol signal interface .................. 880
23.4.6 HyperBus protocol ............................................................... 883
23.4.7 Specific features ................................................................. 887
23.4.8 OCTOSPI operating mode introduction .................................... 888
23.4.9 OCTOSPI indirect mode .......................................................... 888
23.4.10 OCTOSPI automatic status-polling mode ................................. 890
23.4.11 OCTOSPI memory-mapped mode ........................................... 891
23.4.12 OCTOSPI configuration introduction ..................................... 891
23.4.13 OCTOSPI system configuration ............................................. 891
23.4.14 OCTOSPI device configuration .............................................. 892
23.4.15 OCTOSPI regular-command mode configuration ......................... 893
23.4.16 OCTOSPI HyperBus protocol configuration ............................. 895
23.4.17 OCTOSPI error management ................................................ 896
23.4.18 OCTOSPI BUSY and ABORT ............................................... 896
23.4.19 OCTOSPI reconfiguration or deactivation ............................... 897
23.4.20 NCS behavior ................................................................. 897
23.5 Address alignment and data number .......................................... 898
23.6 OCTOSPI interrupts ............................................................... 899
23.7 OCTOSPI registers ............................................................... 900
23.7.1 OCTOSPI control register (OCTOSPI_CR) ................................. 900
23.7.2 OCTOSPI device configuration register 1 (OCTOSPI_DCR1) ....... 903
23.7.3 OCTOSPI device configuration register 2 (OCTOSPI_DCR2) ....... 904
23.7.4 OCTOSPI device configuration register 3 (OCTOSPI_DCR3) ....... 905
23.7.5 OCTOSPI device configuration register 4 (OCTOSPI_DCR4) ....... 905
23.7.6 OCTOSPI status register (OCTOSPI_SR) .................................. 906
23.7.7 OCTOSPI flag clear register (OCTOSPI_FCR) ............................ 907
23.7.8 OCTOSPI data length register (OCTOSPI_DLR) ......................... 907
23.7.9 OCTOSPI address register (OCTOSPI_AR) ............................... 908
23.7.10 OCTOSPI data register (OCTOSPI_DR) .................................. 908
23.7.11 OCTOSPI polling status mask register (OCTOSPI_PSMKR) ....... 909
23.7.12 OCTOSPI polling status match register (OCTOSPI_PSMAR) ....... 910
23.7.13 OCTOSPI polling interval register (OCTOSPI_PIR) .................. 910
23.7.14 OCTOSPI communication configuration register (OCTOSPI_CCR) . 910
23.7.15 OCTOSPI timing configuration register (OCTOSPI_TCR) ........... 912
23.7.16 OCTOSPI instruction register (OCTOSPI_IR) ............................ 913
23.7.17 OCTOSPI alternate bytes register (OCTOSPI_ABR) ............... 913
23.7.18 OCTOSPI low-power timeout register (OCTOSPI_LPTR) .......... 914
23.7.19 OCTOSPI wrap communication configuration register
(OCTOSPI_WPCCR) .................................................. 914
23.7.20 OCTOSPI wrap timing configuration register (OCTOSPI_WPTCR) . 916
23.7.21 OCTOSPI wrap instruction register (OCTOSPI_WPIR) ............. 917
23.7.22 OCTOSPI wrap alternate bytes register (OCTOSPI_WPABR) ..... 917
23.7.23 OCTOSPI write communication configuration register
(OCTOSPI_WCCR) .................................................. 918
23.7.24 OCTOSPI write timing configuration register (OCTOSPI_WTCR) . 920
23.7.25 OCTOSPI write instruction register (OCTOSPI_WIR) ............. 920
23.7.26 OCTOSPI write alternate bytes register (OCTOSPI_WABR) ..... 921
23.7.27 OCTOSPI HyperBus latency configuration register
(OCTOSPI_HLCR) .................................................. 921
23.7.28 OCTOSPI register map .......................................... 922

24 Secure digital input/output MultiMediaCard interface (SDMMC) ........ 925
24.1 SDMMC main features ................................................. 925
24.2 SDMMC implementation ............................................... 925
24.3 SDMMC bus topology .................................................. 926
24.4 SDMMC operation modes .............................................. 928
24.5 SDMMC functional description ...................................... 929
  24.5.1 SDMMC block diagram ............................................ 929
  24.5.2 SDMMC pins and internal signals ................................ 929
  24.5.3 General description ............................................... 930
  24.5.4 SDMMC adapter ................................................... 932
  24.5.5 SDMMC AHB slave interface ..................................... 954
  24.5.6 SDMMC AHB master interface ................................... 954
  24.5.7 AHB and SDMMC_CK clock relation ............................ 958
24.6 Card functional description .......................................... 958
  24.6.1 SD I/O mode ...................................................... 958
  24.6.2 CMD12 send timing ............................................... 966
  24.6.3 Sleep (CMD5) .................................................... 970
  24.6.4 Interrupt mode (Wait-IRQ) ..................................... 971
  24.6.5 Boot operation .................................................. 972
  24.6.6 Response R1b handling .......................................... 975
  24.6.7 Reset and card cycle power .................................... 976
24.7 Hardware flow control ............................................... 977
24.8 Ultra-high-speed phase I (UHS-I) voltage switch ............... 978
24.9 SDMMC interrupts ...................................................... 981
24.10 SDMMC registers ...................................................... 983
  24.10.1 SDMMC power control register (SDMMC_POWER) .......... 983
  24.10.2 SDMMC clock control register (SDMMC_CLKCR) .......... 984
  24.10.3 SDMMC argument register (SDMMC_ARGR) ................. 986
  24.10.4 SDMMC command register (SDMMC_CMDR) ................. 986
  24.10.5 SDMMC command response register (SDMMC_RESPCMDR) ... 988
  24.10.6 SDMMC data timer register (SDMMC_DTIMER) ............ 989
  24.10.7 SDMMC data length register (SDMMC_DLENR) ............ 990
  24.10.8 SDMMC data control register (SDMMC_DCTRL) .......... 991
  24.10.9 SDMMC data counter register (SDMMC_DCNT) ............ 992
  24.10.11 SDMMC status register (SDMMC_STAR) .................. 993
  24.10.12 SDMMC interrupt clear register (SDMMC_ICR) .......... 996
  24.10.13 SDMMC mask register (SDMMC_MASKR) ................. 998
  24.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER) ... 1001
  24.10.15 SDMMC data FIFO registers x (SDMMC_FIFORx) .......... 1001
  24.10.16 SDMMC DMA control register (SDMMC_IDMACR) .......... 1002
  24.10.17 SDMMC IDMA buffer size register (SDMMC_IDMABSIZE) ... 1002
  24.10.18 SDMMC IDMA buffer base address register
            (SDMMC_IDMABASER) ............................................. 1003
  24.10.19 SDMMC IDMA linked list address register (SDMMC_IDMALAR) ... 1003
  24.10.20 SDMMC IDMA linked list memory base register
            (SDMMC_IDMABAR) ............................................. 1004
  24.10.21 SDMMC register map .......................................... 1005

25 Delay block (DLYB) .................................................... 1008
  25.1 Introduction ..................................................... 1008
  25.2 DLYB main features .............................................. 1008
  25.3 DLYB functional description .................................... 1008
    25.3.1 DLYB diagram .............................................. 1008
    25.3.2 DLYB pins and internal signals .......................... 1009
    25.3.3 General description ...................................... 1009
    25.3.4 Delay line length configuration procedure .............. 1010
    25.3.5 Output clock phase configuration procedure .............. 1010
25.4 DLYB registers .................................................. 1011
   25.4.1 DLYB control register (DLYB_CR) ......................... 1011
   25.4.2 DLYB configuration register (DLYB_CFGR) ................. 1012
   25.4.3 DLYB register map ....................................... 1012

26 Analog-to-digital converters (ADC1/2) ............................ 1013
   26.1 Introduction ................................................. 1013
   26.2 ADC main features ......................................... 1013
   26.3 ADC implementation ........................................ 1015
   26.4 ADC functional description ................................. 1016
      26.4.1 ADC block diagram .................................... 1016
      26.4.2 ADC pins and internal signals ......................... 1017
      26.4.3 ADC clocks ............................................. 1020
      26.4.4 ADC connectivity ...................................... 1022
      26.4.5 Slave AHB interface ................................... 1024
      26.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN) ........................................... 1024
      26.4.7 Single-ended and differential input channels .......... 1025
      26.4.8 Calibration (ADCAL, ADCALDIFF, ADC_CALFACT) ........ 1025
      26.4.9 ADC on-off control (ADEN, ADDIS, ADRDY) ............... 1028
      26.4.10 Constraints when writing the ADC control bits ...... 1029
      26.4.11 Channel selection (SQRx, JSQRx) ...................... 1030
      26.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2) .............................................................. 1031
      26.4.13 Single conversion mode (CONT = 0) .................... 1033
      26.4.14 Continuous conversion mode (CONT = 1) ............... 1033
      26.4.15 Starting conversions (ADSTART, JADSTART) .......... 1034
      26.4.16 ADC timing .............................................. 1035
      26.4.17 Stopping an ongoing conversion (ADSTP, JADSTP) .... 1036
      26.4.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN) ................................. 1037
      26.4.19 Injected channel management ............................ 1039
      26.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) ...... 1040
      26.4.21 Queue of context for injected conversions ............ 1041
      26.4.22 Programmable resolution (RES) - fast conversion mode .......................................................... 1049
      26.4.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP) ......................................................... 1050
      26.4.24 End of conversion sequence (EOS, JEOS) ............... 1050
26.4.25 Timing diagrams example (single/continuous modes, hardware/software triggers) ........................................ 1051
26.4.26 Data management ................................................. 1053
26.4.27 Dynamic low-power features ....................................... 1059
26.4.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx) ....... 1064
26.4.29 Oversampler .......................................................... 1068
26.4.30 Dual ADC modes ..................................................... 1075
26.4.31 Temperature sensor .................................................. 1088
26.4.32 VBAT supply monitoring ............................................. 1090
26.4.33 Monitoring the internal voltage reference .................................. 1091
26.4.34 Monitoring the supply voltage ........................................ 1092
26.5 ADC interrupts .......................................................... 1093
26.6 ADC registers (for each ADC) ........................................... 1094
  26.6.1 ADC interrupt and status register (ADC_ISR) ................. 1094
  26.6.2 ADC interrupt enable register (ADC_IER) ...................... 1096
  26.6.3 ADC control register (ADC_CR) ................................. 1098
  26.6.4 ADC configuration register (ADC_CFGR) ...................... 1102
  26.6.5 ADC configuration register 2 (ADC_CFGR2) .................. 1106
  26.6.6 ADC sample time register 1 (ADC_SMPR1) ................... 1109
  26.6.7 ADC sample time register 2 (ADC_SMPR2) ................... 1109
  26.6.8 ADC watchdog threshold register 1 (ADC_TR1) ............. 1110
  26.6.9 ADC watchdog threshold register 2 (ADC_TR2) ............. 1111
  26.6.10 ADC watchdog threshold register 3 (ADC_TR3) ............ 1112
  26.6.11 ADC regular sequence register 1 (ADC_SQR1) ............. 1112
  26.6.12 ADC regular sequence register 2 (ADC_SQR2) ............. 1113
  26.6.13 ADC regular sequence register 3 (ADC_SQR3) ............. 1114
  26.6.14 ADC regular sequence register 4 (ADC_SQR4) ............. 1115
  26.6.15 ADC regular data register (ADC_DR) ....................... 1116
  26.6.16 ADC injected sequence register (ADC_JSQR) .............. 1116
  26.6.17 ADC offset y register (ADC_OFYy) ......................... 1119
  26.6.18 ADC injected channel y data register (ADC_JDRy) .......... 1120
  26.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR) ... 1121
  26.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR) .... 1121
  26.6.21 ADC Differential mode selection register (ADC_DIFSEL) .... 1122
  26.6.22 ADC calibration factors (ADC_CALFACT) ................... 1123
  26.6.23 ADC option register (ADC_OR) ............................. 1123
26.7 ADC common registers ........................................... 1124
  26.7.1 ADC common status register (ADC_CSR) ................. 1124
  26.7.2 ADC common control register (ADC_CCR) ................. 1126
  26.7.3 ADC common regular data register for dual mode (ADC_CDR) ... 1129
  26.7.4 ADC hardware configuration register (ADC_HWCFGR0) .... 1129
  26.7.5 ADC version register (ADC_VERR) ......................... 1130
  26.7.6 ADC identification register (ADC_IPDR) .................. 1130
  26.7.7 ADC size identification register (ADC_SIDR) ............ 1131

26.8 ADC register map ............................................... 1132

27 Digital temperature sensor (DTS) .................................. 1136
  27.1 Introduction .................................................. 1136
  27.2 DTS main features ........................................... 1136
  27.3 DTS functional description .................................. 1137
    27.3.1 DTS block diagram ...................................... 1137
    27.3.2 DTS internal signals .................................. 1137
    27.3.3 DTS block operation ................................... 1138
    27.3.4 Operating modes ....................................... 1138
    27.3.5 Calibration ............................................ 1138
    27.3.6 Prescaler .............................................. 1138
    27.3.7 Temperature measurement principles ..................... 1139
    27.3.8 Sampling time ......................................... 1140
    27.3.9 Quick measurement mode ................................ 1140
    27.3.10 Trigger input ........................................ 1141
    27.3.11 On-off control and ready flag ......................... 1141
    27.3.12 Temperature measurement sequence .................... 1142
  27.4 DTS low-power modes ......................................... 1143
  27.5 DTS interrupts ............................................... 1143
    27.5.1 Temperature window comparator ......................... 1143
    27.5.2 Synchronous interrupt ................................ 1143
    27.5.3 Asynchronous wakeup ................................... 1143
  27.6 DTS registers ................................................ 1144
    27.6.1 Temperature sensor configuration register 1 (DTS_CFR1) .... 1144
    27.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1) ..... 1146
    27.6.3 Temperature sensor ramp value register (DTS_RAMPVALR) .... 1146
    27.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1) ... 1147
<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>27.6.5 Temperature sensor data register (DTS_DR)</td>
</tr>
<tr>
<td>27.6.6 Temperature sensor status register (DTS_SR)</td>
</tr>
<tr>
<td>27.6.7 Temperature sensor interrupt enable register (DTS_ITENR)</td>
</tr>
<tr>
<td>27.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR)</td>
</tr>
<tr>
<td>27.6.9 Temperature sensor option register (DTS_OR)</td>
</tr>
<tr>
<td>27.6.10 DTS register map</td>
</tr>
</tbody>
</table>

28 Digital-to-analog converter (DAC) .......................... 1153

28.1 Introduction .................................. 1153
28.2 DAC main features .............................. 1153
28.3 DAC implementation ............................ 1154
28.4 DAC functional description ...................... 1155
    28.4.1 DAC block diagram ......................... 1155
    28.4.2 DAC pins and internal signals .............. 1156
    28.4.3 DAC clocks .................................. 1157
    28.4.4 DAC channel enable ......................... 1157
    28.4.5 DAC data format ............................ 1157
    28.4.6 DAC conversion ............................. 1159
    28.4.7 DAC output voltage ......................... 1160
    28.4.8 DAC trigger selection ...................... 1160
    28.4.9 DMA requests .............................. 1161
    28.4.10 Noise generation .......................... 1162
    28.4.11 Triangle-wave generation .................. 1163
    28.4.12 DAC channel modes ....................... 1164
    28.4.13 DAC channel buffer calibration .......... 1167
    28.4.14 Dual DAC channel conversion modes (if dual channels are available) ........... 1168
28.5 DAC in low-power modes .......................... 1172
28.6 DAC interrupts ................................ 1173
28.7 DAC registers .................................. 1174
    28.7.1 DAC control register (DAC_CR) ............ 1174
    28.7.2 DAC software trigger register (DAC_SWTRGR) | 1177 |
    28.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1) .......... 1178
    28.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1) .......... 1179
28.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1) ........................................... 1179
28.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2) ................................. 1180
28.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2) ............................... 1180
28.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2) ................................. 1181
28.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD) ........................................... 1181
28.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD) ........................................... 1182
28.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD) ........................................... 1182
28.7.12 DAC channel1 data output register (DAC_DOR1) ................................................................. 1183
28.7.13 DAC channel2 data output register (DAC_DOR2) ................................................................. 1183
28.7.14 DAC status register (DAC_SR) .................. .......................... ........................... .............................. 1184
28.7.15 DAC calibration control register (DAC_CCR) ................................................................. 1185
28.7.16 DAC mode control register (DAC_MCR) ................................................................. 1186
28.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1) ........................................... 1187
28.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2) ........................................... 1188
28.7.19 DAC sample and hold time register (DAC_SHHR) ........................................... 1188
28.7.20 DAC sample and hold refresh time register (DAC_SHRR) ........................................... 1189
28.7.21 DAC register map ................................................................. 1190

29 Voltage reference buffer (VREFBUF) .............................. 1192
29.1 Introduction ................................................................. 1192
29.2 VREFBUF implementation ................................................................. 1192
29.3 VREFBUF functional description ................................................................. 1192
29.4 VREFBUF trimming ................................................................. 1193
29.5 VREFBUF registers ................................................................. 1193
29.5.1 VREFBUF control and status register (VREFBUF_CSR) ................................................................. 1193
29.5.2 VREFBUF calibration control register (VREFBUF_CCR) ................................................................. 1194
29.5.3 VREFBUF register map ................................................................. 1195

30 Digital camera interface (DCMI) .............................. 1196
30.1 Introduction ................................................................. 1196
## 30 DCMI

### 30.2 DCMI main features ........................................... 1196

### 30.3 DCMI functional description ................................ 1196
- 30.3.1 DCMI block diagram .................................. 1196
- 30.3.2 DCMI pins and internal signals .................... 1197
- 30.3.3 DCMI clocks ........................................... 1198
- 30.3.4 DCMI DMA interface .................................. 1198
- 30.3.5 DCMI physical interface .............................. 1198
- 30.3.6 DCMI synchronization ................................ 1200
- 30.3.7 DCMI capture modes ................................ 1202
- 30.3.8 DCMI crop feature .................................. 1203
- 30.3.9 DCMI JPEG format ................................... 1204
- 30.3.10 DCMI FIFO ........................................... 1204
- 30.3.11 DCMI data format description ................. 1205

### 30.4 DCMI interrupts ................................................ 1207

### 30.5 DCMI registers .................................................. 1208
- 30.5.1 DCMI control register (DCMI_CR) .............. 1208
- 30.5.2 DCMI status register (DCMI_SR) ................. 1210
- 30.5.3 DCMI raw interrupt status register (DCMI_RIS) .... 1211
- 30.5.4 DCMI interrupt enable register (DCMI_IER) ... 1212
- 30.5.5 DCMI masked interrupt status register (DCMI_MIS) 1213
- 30.5.6 DCMI interrupt clear register (DCMI_ICR) ...... 1214
- 30.5.7 DCMI embedded synchronization code register (DCMI_ESCR) 1214
- 30.5.8 DCMI embedded synchronization unmask register (DCMI_ESUR) 1215
- 30.5.9 DCMI crop window start (DCMI_CWSRTRT) ........ 1216
- 30.5.10 DCMI crop window size (DCMI_CWSIZE) ....... 1216
- 30.5.11 DCMI data register (DCMI_DR) ................. 1217
- 30.5.12 DCMI register map ................................ 1217

## 31 Parallel synchronous slave interface (PSSI) ................ 1219

### 31.1 Introduction .................................................. 1219

### 31.2 PSSI main features .......................................... 1219

### 31.3 PSSI functional description ................................ 1219
- 31.3.1 PSSI block diagram .................................. 1220
- 31.3.2 PSSI pins and internal signals .................. 1220
- 31.3.3 PSSI clock ............................................ 1221
- 31.3.4 PSSI data management .............................. 1221
<table>
<thead>
<tr>
<th>31.3.5</th>
<th>PSSI optional control signals</th>
<th>1223</th>
</tr>
</thead>
<tbody>
<tr>
<td>31.4</td>
<td>PSSI interrupts</td>
<td>1226</td>
</tr>
<tr>
<td>31.5</td>
<td>PSSI registers</td>
<td>1227</td>
</tr>
<tr>
<td>31.5.1</td>
<td>PSSI control register (PSSI_CR)</td>
<td>1227</td>
</tr>
<tr>
<td>31.5.2</td>
<td>PSSI status register (PSSI_SR)</td>
<td>1228</td>
</tr>
<tr>
<td>31.5.3</td>
<td>PSSI raw interrupt status register (PSSI_RIS)</td>
<td>1229</td>
</tr>
<tr>
<td>31.5.4</td>
<td>PSSI interrupt enable register (PSSI_IER)</td>
<td>1230</td>
</tr>
<tr>
<td>31.5.5</td>
<td>PSSI masked interrupt status register (PSSI_MIS)</td>
<td>1230</td>
</tr>
<tr>
<td>31.5.6</td>
<td>PSSI interrupt clear register (PSSI_ICR)</td>
<td>1231</td>
</tr>
<tr>
<td>31.5.7</td>
<td>PSSI data register (PSSI_DR)</td>
<td>1231</td>
</tr>
<tr>
<td>31.5.8</td>
<td>PSSI register map</td>
<td>1232</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>32</th>
<th>True random number generator (RNG)</th>
<th>1233</th>
</tr>
</thead>
<tbody>
<tr>
<td>32.1</td>
<td>Introduction</td>
<td>1233</td>
</tr>
<tr>
<td>32.2</td>
<td>RNG main features</td>
<td>1233</td>
</tr>
<tr>
<td>32.3</td>
<td>RNG functional description</td>
<td>1234</td>
</tr>
<tr>
<td>32.3.1</td>
<td>RNG block diagram</td>
<td>1234</td>
</tr>
<tr>
<td>32.3.2</td>
<td>RNG internal signals</td>
<td>1234</td>
</tr>
<tr>
<td>32.3.3</td>
<td>Random number generation</td>
<td>1235</td>
</tr>
<tr>
<td>32.3.4</td>
<td>RNG initialization</td>
<td>1238</td>
</tr>
<tr>
<td>32.3.5</td>
<td>RNG operation</td>
<td>1239</td>
</tr>
<tr>
<td>32.3.6</td>
<td>RNG clocking</td>
<td>1240</td>
</tr>
<tr>
<td>32.3.7</td>
<td>Error management</td>
<td>1240</td>
</tr>
<tr>
<td>32.3.8</td>
<td>RNG low-power use</td>
<td>1241</td>
</tr>
<tr>
<td>32.4</td>
<td>RNG interrupts</td>
<td>1242</td>
</tr>
<tr>
<td>32.5</td>
<td>RNG processing time</td>
<td>1243</td>
</tr>
<tr>
<td>32.6</td>
<td>RNG entropy source validation</td>
<td>1243</td>
</tr>
<tr>
<td>32.6.1</td>
<td>Introduction</td>
<td>1243</td>
</tr>
<tr>
<td>32.6.2</td>
<td>Validation conditions</td>
<td>1243</td>
</tr>
<tr>
<td>32.6.3</td>
<td>Data collection</td>
<td>1244</td>
</tr>
<tr>
<td>32.7</td>
<td>RNG registers</td>
<td>1244</td>
</tr>
<tr>
<td>32.7.1</td>
<td>RNG control register (RNG_CR)</td>
<td>1244</td>
</tr>
<tr>
<td>32.7.2</td>
<td>RNG status register (RNG_SR)</td>
<td>1247</td>
</tr>
<tr>
<td>32.7.3</td>
<td>RNG data register (RNG_DR)</td>
<td>1248</td>
</tr>
<tr>
<td>32.7.4</td>
<td>RNG health test control register (RNG_HTCR)</td>
<td>1248</td>
</tr>
<tr>
<td>32.7.5</td>
<td>RNG register map</td>
<td>1249</td>
</tr>
</tbody>
</table>
33 AES hardware accelerator (AES) ........................................... 1250
  33.1 Introduction ......................................................... 1250
  33.2 AES main features .................................................. 1250
  33.3 AES implementation ............................................... 1251
  33.4 AES functional description ....................................... 1251
    33.4.1 AES block diagram ........................................... 1251
    33.4.2 AES internal signals ........................................ 1252
    33.4.3 AES reset and clocks ....................................... 1252
    33.4.4 AES symmetric cipher implementation ...................... 1252
    33.4.5 AES encryption or decryption typical usage ............... 1253
    33.4.6 AES authenticated encryption, decryption, and cipher-based
            message authentication ................................... 1255
    33.4.7 AES ciphertext stealing and data padding .................. 1255
    33.4.8 AES suspend and resume operations .......................... 1256
    33.4.9 AES basic chaining modes (ECB, CBC) ....................... 1256
    33.4.10 AES counter (CTR) mode ..................................... 1260
    33.4.11 AES Galois/counter mode (GCM) ............................. 1263
    33.4.12 AES Galois message authentication code (GMAC) .......... 1267
    33.4.13 AES counter with CBC-MAC (CCM) ......................... 1269
    33.4.14 AES key sharing with secure AES co-processor .......... 1274
    33.4.15 AES data registers and data swapping .................... 1275
    33.4.16 AES key registers ........................................... 1277
    33.4.17 AES initialization vector registers ...................... 1277
    33.4.18 AES error management ...................................... 1278
  33.5 AES interrupts ..................................................... 1279
  33.6 AES DMA requests .................................................. 1279
  33.7 AES processing latency ........................................... 1280
  33.8 AES registers ...................................................... 1282
    33.8.1 AES control register (AES_CR) .............................. 1282
    33.8.2 AES status register (AES_SR) ................................ 1284
    33.8.3 AES data input register (AES_DINR) ....................... 1285
    33.8.4 AES data output register (AES_DOUTR) .................... 1286
    33.8.5 AES key register 0 (AES_KEYR0) ........................... 1286
    33.8.6 AES key register 1 (AES_KEYR1) ........................... 1287
    33.8.7 AES key register 2 (AES_KEYR2) ........................... 1287
    33.8.8 AES key register 3 (AES_KEYR3) ........................... 1287
33.8.9 AES initialization vector register 0 (AES_IVR0) ........................................ 1288
33.8.10 AES initialization vector register 1 (AES_IVR1) ...................................... 1288
33.8.11 AES initialization vector register 2 (AES_IVR2) ...................................... 1288
33.8.12 AES initialization vector register 3 (AES_IVR3) ...................................... 1289
33.8.13 AES key register 4 (AES_KEYR4) ............................................................... 1289
33.8.14 AES key register 5 (AES_KEYR5) ............................................................... 1289
33.8.15 AES key register 6 (AES_KEYR6) ............................................................... 1290
33.8.16 AES key register 7 (AES_KEYR7) ............................................................... 1290
33.8.17 AES suspend registers (AES_SUSPRx) ...................................................... 1290
33.8.18 AES interrupt enable register (AES_IER) .................................................. 1291
33.8.19 AES interrupt status register (AES_ISR) .................................................... 1292
33.8.20 AES interrupt clear register (AES_ICR) ..................................................... 1293
33.8.21 AES register map ...................................................................................... 1293

34 Secure AES coprocessor (SAES) ................................................................. 1296
34.1 Introduction .................................................................................................. 1296
34.2 SAES main features .................................................................................... 1297
34.3 SAES implementation ................................................................................. 1297
34.4 SAES functional description ...................................................................... 1298
  34.4.1 SAES block diagram ............................................................................. 1298
  34.4.2 SAES internal signals .......................................................................... 1298
  34.4.3 SAES reset and clocks ......................................................................... 1299
  34.4.4 SAES symmetric cipher implementation .......................................... 1299
  34.4.5 SAES encryption or decryption typical usage ...................................... 1300
  34.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication ............................................... 1303
  34.4.7 SAES ciphertext stealing and data padding ....................................... 1303
  34.4.8 SAES suspend and resume operations .............................................. 1304
  34.4.9 SAES basic chaining modes (ECB, CBC) ........................................... 1304
  34.4.10 SAES counter (CTR) mode ................................................................. 1308
  34.4.11 SAES Galois/counter mode (GCM) .................................................... 1311
  34.4.12 SAES Galois message authentication code (GMAC) .................... 1314
  34.4.13 SAES counter with CBC-MAC (CCM) .............................................. 1316
  34.4.14 SAES operation with wrapped keys ................................................. 1321
  34.4.15 SAES operation with shared keys ...................................................... 1325
  34.4.16 SAES data registers and data swapping ......................................... 1326
  34.4.17 SAES key registers ........................................................................... 1329
34.4.18 SAES initialization vector registers ............................. 1330
34.4.19 SAES error management ............................................ 1331
34.5 SAES interrupts ......................................................... 1333
34.6 SAES DMA requests ....................................................... 1333
34.7 SAES processing latency ................................................ 1334
34.8 SAES registers ............................................................ 1335
   34.8.1 SAES control register (SAES_CR) .............................. 1335
   34.8.2 SAES status register (SAES_SR) ................................. 1338
   34.8.3 SAES data input register (SAES_DINR) ....................... 1339
   34.8.4 SAES data output register (SAES_DOUTR) ................... 1340
   34.8.5 SAES key register 0 (SAES_KEYR0) ......................... 1340
   34.8.6 SAES key register 1 (SAES_KEYR1) ......................... 1341
   34.8.7 SAES key register 2 (SAES_KEYR2) ......................... 1341
   34.8.8 SAES key register 3 (SAES_KEYR3) ......................... 1341
   34.8.9 SAES initialization vector register 0 (SAES_IVR0) .... 1342
   34.8.10 SAES initialization vector register 1 (SAES_IVR1) .... 1342
   34.8.11 SAES initialization vector register 2 (SAES_IVR2) .... 1342
   34.8.12 SAES initialization vector register 3 (SAES_IVR3) .... 1343
   34.8.13 SAES key register 4 (SAES_KEYR4) ....................... 1343
   34.8.14 SAES key register 5 (SAES_KEYR5) ....................... 1343
   34.8.15 SAES key register 6 (SAES_KEYR6) ....................... 1344
   34.8.16 SAES key register 7 (SAES_KEYR7) ....................... 1344
   34.8.17 SAES suspend registers (SAES_SUSPRx) .................... 1344
   34.8.18 SAES interrupt enable register (SAES_IER) .............. 1345
   34.8.19 SAES interrupt status register (SAES_ISR) .............. 1346
   34.8.20 SAES interrupt clear register (SAES_ICR) .............. 1347
   34.8.21 SAES register map .............................................. 1348

35 Hash processor (HASH) ..................................................... 1350
   35.1 Introduction ......................................................... 1350
   35.2 HASH main features ................................................ 1350
   35.3 HASH implementation ............................................... 1351
   35.4 HASH functional description .................................... 1351
      35.4.1 HASH block diagram ......................................... 1351
      35.4.2 HASH internal signals ...................................... 1351
      35.4.3 About secure hash algorithms ............................. 1352
35.4.4 Message data feeding ........................................ 1352
35.4.5 Message digest computing .................................... 1354
35.4.6 Message padding ............................................... 1355
35.4.7 HMAC operation ................................................ 1357
35.4.8 HASH suspend/resume operations ............................... 1359
35.4.9 HASH DMA interface .......................................... 1361
35.4.10 HASH error management ...................................... 1361
35.5 HASH interrupts .................................................. 1361
35.6 HASH processing time ........................................... 1362
35.7 HASH registers .................................................... 1362
35.7.1 HASH control register (HASH_CR) ............................ 1362
35.7.2 HASH data input register (HASH_DIN) ....................... 1364
35.7.3 HASH start register (HASH_STR) .............................. 1365
35.7.4 HASH digest registers ......................................... 1366
35.7.5 HASH interrupt enable register (HASH_IMR) ............... 1368
35.7.6 HASH status register (HASH_SR) ............................ 1368
35.7.7 HASH context swap registers ................................. 1369
35.7.8 HASH register map ............................................ 1370

36 Public key accelerator (PKA) ...................................... 1372
36.1 Introduction ...................................................... 1372
36.2 PKA main features ............................................... 1372
36.3 PKA functional description ...................................... 1373
36.3.1 PKA block diagram ............................................. 1373
36.3.2 PKA internal signals ......................................... 1373
36.3.3 PKA reset and clocks ......................................... 1373
36.3.4 PKA public key acceleration .................................. 1374
36.3.5 Typical applications for PKA .................................. 1375
36.3.6 PKA procedure to perform an operation ...................... 1378
36.3.7 PKA error management ....................................... 1379
36.4 PKA operating modes ............................................. 1379
36.4.1 Introduction .................................................... 1379
36.4.2 Montgomery parameter computation ......................... 1381
36.4.3 Modular addition ............................................. 1381
36.4.4 Modular subtraction ......................................... 1381
36.4.5 Modular and Montgomery multiplication .................... 1382
36.4.6 Modular exponentiation ............................................. 1383
36.4.7 Modular inversion .................................................. 1384
36.4.8 Modular reduction .................................................. 1385
36.4.9 Arithmetic addition ............................................... 1385
36.4.10 Arithmetic subtraction ............................................ 1385
36.4.11 Arithmetic multiplication ......................................... 1386
36.4.12 Arithmetic comparison ............................................. 1386
36.4.13 RSA CRT exponentiation .......................................... 1387
36.4.14 Point on elliptic curve Fp check ................................. 1387
36.4.15 ECC Fp scalar multiplication ..................................... 1388
36.4.16 ECDSA sign .......................................................... 1389
36.4.17 ECDSA verification ................................................ 1391
36.4.18 ECC complete addition .......................................... 1392
36.4.19 ECC double base ladder .......................................... 1392
36.4.20 ECC projective to affine .......................................... 1393
36.5 Example of configurations and processing times .................. 1394
36.5.1 Supported elliptic curves ......................................... 1394
36.5.2 Computation times .................................................. 1396
36.6 PKA interrupts .......................................................... 1398
36.7 PKA registers .......................................................... 1399
36.7.1 PKA control register (PKA_CR) ................................ 1399
36.7.2 PKA status register (PKA_SR) .................................. 1401
36.7.3 PKA clear flag register (PKA_CLRFR) ......................... 1402
36.7.4 PKA RAM ............................................................. 1402
36.7.5 PKA register map ................................................... 1403
37 On-the-fly decryption engine (OTFDEC) ............................. 1404
37.1 Introduction ............................................................ 1404
37.2 OTFDEC main features ............................................... 1404
37.3 OTFDEC functional description ..................................... 1405
37.3.1 OTFDEC block diagram ......................................... 1405
37.3.2 OTFDEC internal signals ........................................ 1405
37.3.3 OTFDEC on-the-fly decryption ................................... 1406
37.3.4 OTFDEC usage of AES in counter mode decryption .......... 1407
37.3.5 Flow control management ......................................... 1408
37.3.6 OTFDEC error management ....................................... 1408
37.4 OTFDEC interrupts ................................................................. 1409
37.5 OTFDEC application information ........................................ 1409
  37.5.1 OTFDEC initialization process ........................................... 1409
  37.5.2 OTFDEC and power management ......................................... 1411
  37.5.3 Encrypting for OTFDEC .................................................. 1411
  37.5.4 OTFDEC key CRC source code .......................................... 1412
37.6 OTFDEC registers .............................................................. 1413
  37.6.1 OTFDEC control register (OTFDEC_CR) ............................ 1413
  37.6.2 OTFDEC privileged access control configuration register (OTFDEC_PRIVCFGR) .................................................. 1414
  37.6.3 OTFDEC region x configuration register (OTFDEC_RxCFGR) .......... 1414
  37.6.4 OTFDEC region x start address register (OTFDEC_RxSTARTADDR) .... 1416
  37.6.5 OTFDEC region x end address register (OTFDEC_RxENDADDR) ....... 1416
  37.6.6 OTFDEC region x nonce register 0 (OTFDEC_RxNONCER0) .......... 1417
  37.6.7 OTFDEC region x nonce register 1 (OTFDEC_RxNONCER1) .......... 1418
  37.6.8 OTFDEC region x key register 0 (OTFDEC_RxKEYR0) ............... 1418
  37.6.9 OTFDEC region x key register 1 (OTFDEC_RxKEYR1) ............... 1419
  37.6.10 OTFDEC region x key register 2 (OTFDEC_RxKEYR2) .............. 1419
  37.6.11 OTFDEC region x key register 3 (OTFDEC_RxKEYR3) .............. 1420
  37.6.12 OTFDEC interrupt status register (OTFDEC_ISR) ................ 1420
  37.6.13 OTFDEC interrupt clear register (OTFDEC_ICR) .................. 1421
  37.6.14 OTFDEC interrupt enable register (OTFDEC_IER) ................. 1422
  37.6.15 OTFDEC register map .................................................... 1423

38 Advanced-control timers (TIM1/TIM8) ...................................... 1427
  38.1 TIM1/TIM8 introduction .................................................... 1427
  38.2 TIM1/TIM8 main features ................................................ 1427
  38.3 TIM1/TIM8 functional description ....................................... 1428
    38.3.1 Block diagram ............................................................ 1428
    38.3.2 TIM1/TIM8 pins and internal signals ................................ 1429
    38.3.3 Time-base unit ........................................................... 1433
    38.3.4 Counter modes ............................................................ 1435
    38.3.5 Repetition counter ....................................................... 1447
    38.3.6 External trigger input ................................................ 1448
    38.3.7 Clock selection ........................................................... 1449
    38.3.8 Capture/compare channels ............................................. 1453
38.3.9 Input capture mode ............................................. 1456
38.3.10 PWM input mode ............................................. 1457
38.3.11 Forced output mode ......................................... 1458
38.3.12 Output compare mode ....................................... 1458
38.3.13 PWM mode .................................................... 1460
38.3.14 Asymmetric PWM mode .................................... 1468
38.3.15 Combined PWM mode ....................................... 1469
38.3.16 Combined 3-phase PWM mode ............................. 1470
38.3.17 Complementary outputs and dead-time insertion ....... 1471
38.3.18 Using the break function ................................... 1474
38.3.19 Bidirectional break inputs .................................. 1480
38.3.20 Clearing the tim_ocxref signal on an external event   1481
38.3.21 6-step PWM generation ...................................... 1483
38.3.22 One-pulse mode .............................................. 1484
38.3.23 Retriggerable One-pulse mode ............................. 1486
38.3.24 Pulse on compare mode ..................................... 1487
38.3.25 Encoder interface mode ..................................... 1489
38.3.26 Direction bit output ......................................... 1507
38.3.27 UIF bit remapping .......................................... 1508
38.3.28 Timer input XOR function ................................. 1508
38.3.29 Interfacing with Hall sensors .............................. 1508
38.3.30 Timer synchronization ...................................... 1510
38.3.31 ADC triggers ................................................ 1515
38.3.32 DMA burst mode ............................................ 1515
38.3.33 TIM1/TIM8 DMA requests ................................... 1516
38.3.34 Debug mode .................................................. 1516
38.4 TIM1/TIM8 low-power modes .................................. 1517
38.5 TIM1/TIM8 interrupts ............................................ 1517
38.6 TIM1/TIM8 registers ............................................. 1518
38.6.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8) ......... 1518
38.6.2 TIMx control register 2 (TIMx_CR2)(x = 1, 8) ......... 1519
38.6.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8) 1523
38.6.4 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8) 1527
38.6.5 TIMx status register (TIMx_SR)(x = 1, 8) ............... 1528
38.6.6 TIMx event generation register (TIMx_EGR)(x = 1, 8) ... 1531
38.6.7 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 1, 8) ................................................ 1532
38.6.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8) .............................................. 1534
38.6.9 TIMx capture/compare mode register 2 (TIMx_CCMR2) (x = 1, 8) ..................................................... 1537
38.6.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8) .......................................... 1538
38.6.11 TIMx capture/compare enable register (TIMx_CCR)(x = 1, 8) .......................................................... 1541
38.6.12 TIMx counter (TIMx_CNT)(x = 1, 8) ................................................................................................. 1545
38.6.13 TIMx prescaler (TIMx_PSC)(x = 1, 8) ................................................................................................. 1545
38.6.14 TIMx auto-reload register (TIMx_ARR)(x = 1, 8) .......................................................... 1546
38.6.15 TIMx repetition counter register (TIMx_RCR)(x = 1, 8) .......................................................... 1546
38.6.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8) .......................................................... 1547
38.6.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8) .......................................................... 1547
38.6.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8) .......................................................... 1548
38.6.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8) .......................................................... 1549
38.6.20 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8) .................................................. 1550
38.6.21 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8) .......................................................... 1554
38.6.22 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8) .......................................................... 1555
38.6.23 TIMx capture/compare mode register 3 (TIMx_CCMR3) (x = 1, 8) .................................................. 1556
38.6.24 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 1, 8) .......................................................... 1557
38.6.25 TIMx timer encoder control register (TIMx_ECR)(x = 1, 8) .......................................................... 1558
38.6.26 TIMx timer input selection register (TIMx_TISEL)(x = 1, 8) .................................................. 1559
38.6.27 TIMx alternate function option register 1 (TIMx_AF1)(x = 1, 8) .................................................. 1560
38.6.28 TIMx alternate function register 2 (TIMx_AF2)(x = 1, 8) .......................................................... 1563
38.6.29 TIMx DMA control register (TIMx_DCR)(x = 1, 8) .......................................................... 1565
38.6.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8) .................................................. 1567
38.6.31 TIMx register map ................................................. 1567

39 General-purpose timers (TIM2/TIM3/TIM4/TIM5) ......................... 1570
39.1 TIM2/TIM3/TIM4/TIM5 introduction ........................................ 1570
39.2 TIM2/TIM3/TIM4/TIM5 main features ........................................ 1570
39.3 TIM2/TIM3/TIM4/TIM5 implementation ........................................ 1571
39.4 TIM2/TIM3/TIM4/TIM5 functional description ................................. 1572
   39.4.1 Block diagram .......................................................... 1572
   39.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals .................. 1573
   39.4.3 Time-base unit .......................................................... 1576
   39.4.4 Counter modes .......................................................... 1578
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>39.4.5</td>
<td>Clock selection</td>
<td>1590</td>
</tr>
<tr>
<td>39.4.6</td>
<td>Capture/compare channels</td>
<td>1594</td>
</tr>
<tr>
<td>39.4.7</td>
<td>Input capture mode</td>
<td>1596</td>
</tr>
<tr>
<td>39.4.8</td>
<td>PWM input mode</td>
<td>1597</td>
</tr>
<tr>
<td>39.4.9</td>
<td>Forced output mode</td>
<td>1598</td>
</tr>
<tr>
<td>39.4.10</td>
<td>Output compare mode</td>
<td>1598</td>
</tr>
<tr>
<td>39.4.11</td>
<td>PWM mode</td>
<td>1600</td>
</tr>
<tr>
<td>39.4.12</td>
<td>Asymmetric PWM mode</td>
<td>1608</td>
</tr>
<tr>
<td>39.4.13</td>
<td>Combined PWM mode</td>
<td>1609</td>
</tr>
<tr>
<td>39.4.14</td>
<td>Clearing the tim_ocxref signal on an external event</td>
<td>1610</td>
</tr>
<tr>
<td>39.4.15</td>
<td>One-pulse mode</td>
<td>1612</td>
</tr>
<tr>
<td>39.4.16</td>
<td>Retriggerable one-pulse mode</td>
<td>1613</td>
</tr>
<tr>
<td>39.4.17</td>
<td>Pulse on compare mode</td>
<td>1614</td>
</tr>
<tr>
<td>39.4.18</td>
<td>Encoder interface mode</td>
<td>1616</td>
</tr>
<tr>
<td>39.4.19</td>
<td>Direction bit output</td>
<td>1634</td>
</tr>
<tr>
<td>39.4.20</td>
<td>UIF bit remapping</td>
<td>1635</td>
</tr>
<tr>
<td>39.4.21</td>
<td>Timer input XOR function</td>
<td>1635</td>
</tr>
<tr>
<td>39.4.22</td>
<td>Timers and external trigger synchronization</td>
<td>1635</td>
</tr>
<tr>
<td>39.4.23</td>
<td>Timer synchronization</td>
<td>1639</td>
</tr>
<tr>
<td>39.4.24</td>
<td>ADC triggers</td>
<td>1644</td>
</tr>
<tr>
<td>39.4.25</td>
<td>DMA burst mode</td>
<td>1645</td>
</tr>
<tr>
<td>39.4.26</td>
<td>TIM2/TIM3/TIM4/TIM5 DMA requests</td>
<td>1646</td>
</tr>
<tr>
<td>39.4.27</td>
<td>Debug mode</td>
<td>1646</td>
</tr>
<tr>
<td>39.4.28</td>
<td>TIM2/TIM3/TIM4/TIM5 low-power modes</td>
<td>1646</td>
</tr>
<tr>
<td>39.4.29</td>
<td>TIM2/TIM3/TIM4/TIM5 interrupts</td>
<td>1647</td>
</tr>
<tr>
<td>39.5</td>
<td>TIM2/TIM3/TIM4/TIM5 registers</td>
<td>1648</td>
</tr>
<tr>
<td>39.5.1</td>
<td>TIMx control register 1 (TIMx_CR1)</td>
<td>1648</td>
</tr>
<tr>
<td>39.5.2</td>
<td>TIMx control register 2 (TIMx_CR2)</td>
<td>1649</td>
</tr>
<tr>
<td>39.5.3</td>
<td>TIMx slave mode control register (TIMx_SMCR)</td>
<td>1651</td>
</tr>
<tr>
<td>39.5.4</td>
<td>TIMx DMA/Interrupt enable register (TIMx_DIER)</td>
<td>1655</td>
</tr>
<tr>
<td>39.5.5</td>
<td>TIMx status register (TIMx_SR)</td>
<td>1656</td>
</tr>
<tr>
<td>39.5.6</td>
<td>TIMx event generation register (TIMx_EGR)</td>
<td>1658</td>
</tr>
<tr>
<td>39.5.7</td>
<td>TIMx capture/compare mode register 1 (TIMx_CCMR1)</td>
<td>1659</td>
</tr>
<tr>
<td>39.5.8</td>
<td>TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)</td>
<td>1661</td>
</tr>
<tr>
<td>39.5.9</td>
<td>TIMx capture/compare mode register 2 (TIMx_CCMR2)</td>
<td>1663</td>
</tr>
</tbody>
</table>
40 Basic timers (TIM6/TIM7) ........................................ 1687
  40.1 TIM6/TIM7 introduction .................................. 1687
  40.2 TIM6/TIM7 main features .................................. 1687
  40.3 TIM6/TIM7 functional description ..................... 1688
    40.3.1 TIM6/TIM7 block diagram ......................... 1688
    40.3.2 TIM6/TIM7 internal signals .................... 1688
    40.3.3 TIM6/TIM7 clocks ................................ 1689
    40.3.4 Time-base unit .................................. 1689
    40.3.5 Counting mode .................................. 1691
    40.3.6 UIF bit remapping .............................. 1698
    40.3.7 ADC triggers .................................. 1699
    40.3.8 TIM6/TIM7 DMA requests ....................... 1699
    40.3.9 Debug mode .................................. 1699

40.3.10 TIM6/TIM7 low-power modes .................................................. 1699
40.3.11 TIM6/TIM7 interrupts .............................................................. 1699
40.4 TIM6/TIM7 registers ................................................................. 1700
  40.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7) ....................... 1700
  40.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7) .................... 1702
  40.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7) ... 1702
  40.4.4 TIMx status register (TIMx_SR)(x = 6 to 7) .......................... 1703
  40.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7) .......... 1703
  40.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7) .................................. 1703
  40.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7) .............................. 1704
  40.4.8 TIMx auto-reload register (TIMx_ARR)(x = 6 to 7) .............. 1704
  40.4.9 TIMx register map ............................................................. 1705

41 General-purpose timers (TIM12/TIM13/TIM14) .......................... 1706
  41.1 TIM12/TIM13/TIM14 introduction ............................................ 1706
  41.2 TIM12 main features ............................................................. 1706
  41.3 TIM13/TIM14 main features .................................................... 1707
  41.4 TIM12/TIM13/TIM14 functional description ............................... 1708
    41.4.1 Block diagram ............................................................. 1708
    41.4.2 TIM12/TIM13/TIM14 pins and internal signals ....................... 1709
    41.4.3 Time-base unit ............................................................ 1712
    41.4.4 Counter modes ............................................................ 1714
    41.4.5 Clock selection ........................................................... 1717
    41.4.6 Capture/compare channels .............................................. 1719
    41.4.7 Input capture mode ....................................................... 1721
    41.4.8 PWM input mode (TIM12 only) ......................................... 1722
    41.4.9 Forced output mode ...................................................... 1723
    41.4.10 Output compare mode .................................................... 1724
    41.4.11 PWM mode ................................................................. 1725
    41.4.12 Combined PWM mode (TIM12 only) ................................... 1730
    41.4.13 One-pulse mode .......................................................... 1731
    41.4.14 Retriggerable one pulse mode (TIM12 only) ....................... 1733
    41.4.15 UIF bit remapping ....................................................... 1734
    41.4.16 Timer input XOR function ............................................. 1734
    41.4.17 TIM12 external trigger synchronization .............................. 1734
    41.4.18 Slave mode – combined reset + trigger mode ..................... 1737
    41.4.19 Slave mode – combined reset + gated mode .......................... 1737
41.4.20 Timer synchronization (TIM12 only) ......................................................... 1738
41.4.21 Using timer output as trigger for other timers (TIM13/TIM14 only) ............ 1738
41.4.22 ADC triggers (TIM12 only) ................................................................. 1739
41.4.23 Debug mode ......................................................................................... 1739
41.5 TIM12/TIM13/TIM14 low-power modes ..................................................... 1739
41.6 TIM12/TIM13/TIM14 interrupts ................................................................. 1739
41.7 TIM12 registers ......................................................................................... 1740
  41.7.1 TIM12 control register 1 (TIM12_CR1) ................................................... 1740
  41.7.2 TIM12 control register 2 (TIM12_CR2) ................................................... 1741
  41.7.3 TIM12 slave mode control register (TIM12_SMCR) ............................. 1742
  41.7.4 TIM12 Interrupt enable register (TIM12_DIER) ................................. 1744
  41.7.5 TIM12 status register (TIM12_SR) ....................................................... 1745
  41.7.6 TIM12 event generation register (TIM12_EGR) ................................ 1746
  41.7.7 TIM12 capture/compare mode register 1 (TIM12_CCMR1) ............... 1747
  41.7.8 TIM12 capture/compare mode register 1 [alternate] (TIM12_CCMR1) .... 1748
  41.7.9 TIM12 capture/compare enable register (TIM12_CCER) .................... 1751
  41.7.10 TIM12 counter (TIM12_CNT) ............................................................ 1752
  41.7.11 TIM12 prescaler (TIM12_PSC) .......................................................... 1753
  41.7.12 TIM12 auto-reload register (TIM12_ARR) ......................................... 1753
  41.7.13 TIM12 capture/compare register 1 (TIM12_CCR1) ......................... 1754
  41.7.14 TIM12 capture/compare register 2 (TIM12_CCR2) ......................... 1754
  41.7.15 TIM12 timer input selection register (TIM12_TISEL) ..................... 1755
  41.7.16 TIM12 register map ............................................................................ 1756
41.8 TIM13/TIM14 registers ............................................................................. 1758
  41.8.1 TIMx control register 1 (TIMx_CR1)(x = 13, 14) ............................... 1758
  41.8.2 TIMx Interrupt enable register (TIMx_DIER)(x = 13, 14) ................ 1759
  41.8.3 TIMx status register (TIMx_SR)(x = 13, 14) .................................... 1759
  41.8.4 TIMx event generation register (TIMx_EGR)(x = 13, 14) ............... 1760
  41.8.5 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 13, 14) ... 1761
  41.8.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 13, 14) ...... 1762
  41.8.7 TIMx capture/compare enable register (TIMx_CCER)(x = 13, 14) ...... 1764
  41.8.8 TIMx counter (TIMx_CNT)(x = 13, 14) ............................................. 1765
  41.8.9 TIMx prescaler (TIMx_PSC)(x = 13, 14) ......................................... 1766
42 General purpose timers (TIM15/TIM16/TIM17) .......................... 1770

42.1 TIM15/TIM16/TIM17 introduction ........................................ 1770
42.2 TIM15 main features .......................................................... 1770
42.3 TIM16/TIM17 main features ................................................. 1771
42.4 TIM15/TIM16/TIM17 functional description ............................. 1772
  42.4.1 Block diagram ............................................................. 1772
  42.4.2 TIM15/TIM16/TIM17 pins and internal signals .................... 1773
  42.4.3 Time-base unit .......................................................... 1776
  42.4.4 Counter modes .......................................................... 1778
  42.4.5 Repetition counter ...................................................... 1782
  42.4.6 Clock selection ......................................................... 1783
  42.4.7 Capture/compare channels ............................................ 1785
  42.4.8 Input capture mode ..................................................... 1787
  42.4.9 PWM input mode (only for TIM15) .................................. 1789
  42.4.10 Forced output mode ................................................... 1790
  42.4.11 Output compare mode ................................................. 1790
  42.4.12 PWM mode ............................................................. 1792
  42.4.13 Combined PWM mode (TIM15 only) ................................ 1797
  42.4.14 Complementary outputs and dead-time insertion ................ 1798
  42.4.15 Using the break function ............................................ 1801
  42.4.16 Bidirectional break input .......................................... 1805
  42.4.17 Clearing the tim_ocxref signal on an external event .......... 1806
  42.4.18 6-step PWM generation .............................................. 1807
  42.4.19 One-pulse mode ....................................................... 1809
  42.4.20 Retriggerable one pulse mode (TIM15 only) ...................... 1810
  42.4.21 UIF bit remapping .................................................... 1811
  42.4.22 Timer input XOR function (TIM15 only) .......................... 1811
  42.4.23 External trigger synchronization (TIM15 only) ................ 1811
  42.4.24 Slave mode – combined reset + trigger mode (TIM15 only) .... 1814
  42.4.25 Slave mode – combined reset + gated mode (TIM15 only) ...... 1814
  42.4.26 Timer synchronization (TIM15 only) .............................. 1815
  42.4.27 Using timer output as trigger for other timers (TIM16/TIM17 only) . 1815
42.4.28 ADC triggers (TIM15 only) ........................................... 1815
42.4.29 DMA burst mode ......................................................... 1815
42.4.30 TIM15/TIM16/TIM17 DMA requests ................................. 1816
42.4.31 Debug mode ............................................................. 1816
42.5 TIM15/TIM16/TIM17 low-power modes ................................. 1817
42.6 TIM15/TIM16/TIM17 interrupts .......................................... 1817
42.7 TIM15 registers .............................................................. 1818
  42.7.1 TIM15 control register 1 (TIM15_CR1) ............................ 1818
  42.7.2 TIM15 control register 2 (TIM15_CR2) ............................ 1819
  42.7.3 TIM15 slave mode control register (TIM15_SMCR) ............... 1821
  42.7.4 TIM15 DMA/interrupt enable register (TIM15_DIER) .......... 1823
  42.7.5 TIM15 status register (TIM15_SR) ................................ 1824
  42.7.6 TIM15 event generation register (TIM15_EGR) .................. 1826
  42.7.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1) .... 1827
  42.7.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) ............................................... 1828
  42.7.9 TIM15 capture/compare enable register (TIM15_CCER) ....... 1831
  42.7.10 TIM15 counter (TIM15_CNT) .................................... 1834
  42.7.11 TIM15 prescaler (TIM15_PSC) .................................. 1834
  42.7.12 TIM15 auto-reload register (TIM15_ARR) ....................... 1835
  42.7.13 TIM15 repetition counter register (TIM15_RCR) ............... 1835
  42.7.14 TIM15 capture/compare register 1 (TIM15_CCR1) ............ 1836
  42.7.15 TIM15 capture/compare register 2 (TIM15_CCR2) ............ 1837
  42.7.16 TIM15 break and dead-time register (TIM15_BDTR) .......... 1837
  42.7.17 TIM15 timer deadtime register 2 (TIM15_DTR2) .............. 1840
  42.7.18 TIM15 input selection register (TIM15_TISEL) ................ 1841
  42.7.19 TIM15 alternate function register 1 (TIM15_AF1) ........... 1842
  42.7.20 TIM15 alternate function register 2 (TIM15_AF2) .......... 1844
  42.7.21 TIM15 DMA control register (TIM15_DCR) ..................... 1845
  42.7.22 TIM15 DMA address for full transfer (TIM15_DMAR) ......... 1846
  42.7.23 TIM15 register map ................................................. 1846
42.8 TIM16/TIM17 registers .................................................. 1849
  42.8.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17) ............ 1849
  42.8.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17) .......... 1850
  42.8.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17) .... 1851
  42.8.4 TIMx status register (TIMx_SR)(x = 16 to 17) ............... 1852
  42.8.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17) .... 1853
42.8.6 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 16 to 17) ........................................ 1854
42.8.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17) ................................ 1855
42.8.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) ........................................... 1857
42.8.9 TIMx counter (TIMx_CNT)(x = 16 to 17) .................................................................................. 1860
42.8.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17) .............................................................................. 1860
42.8.11 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17) ......................................................... 1861
42.8.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17) ................................................. 1861
42.8.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17) ............................................ 1862
42.8.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) ........................................ 1863
42.8.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17) .............................................. 1866
42.8.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17) ................................................. 1867
42.8.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17) ............................................. 1867
42.8.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17) ............................................. 1870
42.8.19 TIMx DMA control register (TIMx_DCR)(x = 16 to 17) .................................................. 1870
42.8.20 TIM16/TIM17 DMA address for full transfer (TIMx_DMAR)(x = 16 to 17) .................... 1871
42.8.21 TIM16/TIM17 register map ................................................................................................. 1873

43 Low-power timer (LPTIM) ................................................................. 1875
43.1 Introduction .................................................................................... 1875
43.2 LPTIM main features ...................................................................... 1875
43.3 LPTIM implementation .................................................................... 1876
43.4 LPTIM functional description .......................................................... 1877
  43.4.1 LPTIM block diagram ................................................................. 1877
  43.4.2 LPTIM pins and internal signals .................................................. 1878
  43.4.3 LPTIM input and trigger mapping ................................................ 1880
  43.4.4 LPTIM reset and clocks ............................................................... 1881
  43.4.5 Glitch filter .................................................................................. 1882
  43.4.6 Prescaler ..................................................................................... 1882
  43.4.7 Trigger multiplexer ...................................................................... 1883
  43.4.8 Operating mode ........................................................................... 1883
  43.4.9 Timeout function ......................................................................... 1885
  43.4.10 Waveform generation ................................................................. 1885
  43.4.11 Register update ........................................................................... 1887
  43.4.12 Counter mode ............................................................................. 1887
  43.4.13 Timer enable .............................................................................. 1888
43.4.14 Timer counter reset .............................................. 1888
43.4.15 Encoder mode ................................................. 1889
43.4.16 Repetition Counter ........................................... 1890
43.4.17 Capture/compare channels ................................. 1892
43.4.18 Input capture mode ........................................... 1892
43.4.19 PWM mode ..................................................... 1894
43.4.20 DMA requests ................................................ 1896
43.4.21 Debug mode .................................................... 1897
43.5 LPTIM low-power modes .......................................... 1897
43.6 LPTIM interrupts ................................................ 1897
43.7 LPTIM registers .................................................. 1898
  43.7.1 LPTIM4 interrupt and status register (LPTIM4_ISR) ........ 1899
  43.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR)
      (x = 1 to 3, 5, 6) .............................................. 1900
  43.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR)
      (x = 1 to 3, 5, 6) .............................................. 1902
  43.7.4 LPTIM4 interrupt clear register (LPTIM4_ICR) ............. 1904
  43.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR)
      (x = 1 to 3, 5, 6) .............................................. 1905
  43.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR)
      (x = 1 to 3, 5, 6) .............................................. 1906
  43.7.7 LPTIM4 interrupt enable register (LPTIM4_DIER) ........... 1907
  43.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER)
      (x = 1 to 3, 5, 6) .............................................. 1909
  43.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER)
      (x = 1 to 3, 5, 6) .............................................. 1910
  43.7.10 LPTIM configuration register (LPTIM_CFGR) ............... 1912
  43.7.11 LPTIM control register (LPTIM_CR) ........................ 1915
  43.7.12 LPTIM compare register 1 (LPTIM_CCR1) ................... 1916
  43.7.13 LPTIM autoreload register (LPTIM_ARR) ................... 1917
  43.7.14 LPTIM counter register (LPTIM_CNT) ...................... 1917
  43.7.15 LPTIM configuration register 2 (LPTIM_CFRG2) .......... 1918
  43.7.16 LPTIM repetition register (LPTIM_RCR) ................... 1919
  43.7.17 LPTIM capture/compare mode register 1 (LPTIM_CCMR1) .... 1919
  43.7.18 LPTIM compare register 2 (LPTIM_CCR2) ................... 1922
  43.7.19 LPTIM register map ....................................... 1922

44 Independent watchdog (IWDG) .................................... 1925
44.1 Introduction ................................................. 1925
44.2 IWDG main features ........................................ 1925
44.3 IWDG implementation ....................................... 1925
44.4 IWDG functional description ................................. 1926
  44.4.1 IWDG block diagram ................................. 1926
  44.4.2 IWDG internal signals ............................... 1927
  44.4.3 Software and hardware watchdog modes .............. 1927
  44.4.4 Window option ..................................... 1928
  44.4.5 Debug ............................................ 1931
  44.4.6 Register access protection ......................... 1931
44.5 IWDG low-power modes .................................... 1931
44.6 IWDG interrupts .......................................... 1931
44.7 IWDG registers ............................................ 1933
  44.7.1 IWDG key register (IWDG_KR) ....................... 1934
  44.7.2 IWDG prescaler register (IWDG_PR) .................... 1934
  44.7.3 IWDG reload register (IWDG_RLR) ..................... 1935
  44.7.4 IWDG status register (IWDG_SR) ...................... 1935
  44.7.5 IWDG window register (IWDG_WINR) ................... 1937
  44.7.6 IWDG early wake-up interrupt register (IWDG_EWCR) 1937
  44.7.7 IWDG register map ................................ 1939

45 System window watchdog (WWDG) .............................. 1940
45.1 Introduction .............................................. 1940
45.2 WWDG main features ....................................... 1940
45.3 WWDG implementation ..................................... 1940
45.4 WWDG functional description ............................... 1941
  45.4.1 WWDG block diagram ................................ 1941
  45.4.2 WWDG internal signals .............................. 1941
  45.4.3 Enabling the watchdog ................................ 1942
  45.4.4 Controlling the down-counter ......................... 1942
  45.4.5 How to program the watchdog timeout ............... 1942
  45.4.6 Debug mode ...................................... 1943
45.5 WWDG interrupts .......................................... 1944
45.6 WWDG registers ............................................ 1944
  45.6.1 WWDG control register (WWDG_CR) ................. 1944
  45.6.2 WWDG configuration register (WWDG_CFR) .......... 1945
46 Real-time clock (RTC) .................................................. 1947

46.1 Introduction .......................................................... 1947
46.2 RTC main features ..................................................... 1947
46.3 RTC functional description .......................................... 1948
  46.3.1 RTC block diagram .............................................. 1948
  46.3.2 RTC pins and internal signals ................................ 1950
  46.3.3 GPIOs controlled by the RTC and TAMP ..................... 1951
  46.3.4 RTC secure protection modes .................................. 1955
  46.3.5 RTC privilege protection modes ................................. 1956
  46.3.6 Clock and prescalers ........................................... 1957
  46.3.7 Real-time clock and calendar .................................. 1959
  46.3.8 Calendar ultra-low power mode ................................. 1959
  46.3.9 Programmable alarms ........................................... 1960
  46.3.10 Periodic auto-wake-up ......................................... 1960
  46.3.11 RTC initialization and configuration ......................... 1961
  46.3.12 Reading the calendar .......................................... 1963
  46.3.13 Resetting the RTC ............................................. 1964
  46.3.14 RTC synchronization ........................................... 1965
  46.3.15 RTC reference clock detection ................................. 1965
  46.3.16 RTC smooth digital calibration ............................... 1966
  46.3.17 Timestamp function ........................................... 1968
  46.3.18 Calibration clock output ..................................... 1969
  46.3.19 Tamper and alarm output ..................................... 1969

46.4 RTC low-power modes ............................................... 1970
46.5 RTC interrupts ..................................................... 1970
46.6 RTC registers ....................................................... 1972
  46.6.1 RTC time register (RTC_TR) .................................... 1972
  46.6.2 RTC date register (RTC_DR) .................................... 1973
  46.6.3 RTC subsecond register (RTC_SSR) ............................ 1974
  46.6.4 RTC initialization control and status register (RTC_ICSR) 1975
  46.6.5 RTC prescaler register (RTC_PRER) ............................ 1977
  46.6.6 RTC wake-up timer register (RTC_WUTR) ....................... 1978
  46.6.7 RTC control register (RTC_CR) ................................ 1978
47.1 Introduction ................................................. 2005
47.2 TAMP main features ..................................... 2006
47.3 TAMP functional description .......................... 2007
  47.3.1 TAMP block diagram ............................... 2007
  47.3.2 TAMP pins and internal signals .................. 2008
  47.3.3 GPIOs controlled by the RTC and TAMP ........ 2011
  47.3.4 TAMP register write protection ................. 2011
  47.3.5 TAMP secure protection modes .................... 2011
  47.3.6 Backup registers protection zones ............... 2012
  47.3.7 TAMP privilege protection modes ................. 2012
  47.3.8 Boot hardware key (BHK) .......................... 2013
  47.3.9 Tamper detection .................................... 2013
  47.3.10 TAMP backup registers and other device secrets erase 2013
  47.3.11 Tamper detection configuration and initialization 2015
47.4 TAMP low-power modes .................................. 2020
47.5 TAMP interrupts .................................................. 2021
47.6 TAMP registers .................................................. 2021
  47.6.1 TAMP control register 1 (TAMP_CR1) ....................... 2021
  47.6.2 TAMP control register 2 (TAMP_CR2) ....................... 2023
  47.6.3 TAMP control register 3 (TAMP_CR3) ....................... 2026
  47.6.4 TAMP filter control register (TAMP_FLTCR) ................ 2027
  47.6.5 TAMP active tamper control register 1 (TAMP_ATCR1) .... 2028
  47.6.6 TAMP active tamper seed register (TAMP_ATSEEDR) ....... 2031
  47.6.7 TAMP active tamper output register (TAMP_ATOR) ....... 2032
  47.6.8 TAMP active tamper control register 2 (TAMP_ATCR2) .... 2032
  47.6.9 TAMP secure configuration register (TAMP_SECCFGR) .... 2035
  47.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR) 2037
  47.6.11 TAMP interrupt enable register (TAMP_IER) .............. 2038
  47.6.12 TAMP status register (TAMP_SR) ......................... 2040
  47.6.13 TAMP nonsecure masked interrupt status register (TAMP_MISR) 2042
  47.6.14 TAMP secure masked interrupt status register (TAMP_SMISR) 2043
  47.6.15 TAMP status clear register (TAMP_SCR) .................. 2045
  47.6.16 TAMP monotonic counter 1 register (TAMP_COUNT1R) .... 2047
  47.6.17 TAMP option register (TAMP_OR) ....................... 2047
  47.6.18 TAMP resources protection configuration register (TAMP_RPCFGR) 2048
  47.6.19 TAMP backup x register (TAMP_BKPxR) .................. 2049
  47.6.20 TAMP register map ....................................... 2050

48 Inter-integrated circuit (I2C) interface .......................... 2052
  48.1 Introduction ................................................. 2052
  48.2 I2C main features ........................................... 2052
  48.3 I2C implementation ......................................... 2053
  48.4 I2C functional description .................................. 2053
    48.4.1 I2C block diagram ....................................... 2054
    48.4.2 I2C pins and internal signals .......................... 2055
    48.4.3 I2C clock requirements .................................. 2055
    48.4.4 Mode selection ......................................... 2055
    48.4.5 I2C initialization ....................................... 2056
    48.4.6 Software reset .......................................... 2061
    48.4.7 Data transfer ............................................ 2062
    48.4.8 I2C slave mode ......................................... 2064
## 48 I2C

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>48.4.9</td>
<td>I2C master mode</td>
<td>2073</td>
</tr>
<tr>
<td>48.4.10</td>
<td>I2C_TIMINGR register configuration examples</td>
<td>2084</td>
</tr>
<tr>
<td>48.4.11</td>
<td>SMBus specific features</td>
<td>2085</td>
</tr>
<tr>
<td>48.4.12</td>
<td>SMBus initialization</td>
<td>2088</td>
</tr>
<tr>
<td>48.4.13</td>
<td>SMBus: I2C_TIMEOUTR register configuration examples</td>
<td>2090</td>
</tr>
<tr>
<td>48.4.14</td>
<td>SMBus slave mode</td>
<td>2090</td>
</tr>
<tr>
<td>48.4.15</td>
<td>Wake-up from Stop mode on address match</td>
<td>2098</td>
</tr>
<tr>
<td>48.4.16</td>
<td>Error conditions</td>
<td>2098</td>
</tr>
<tr>
<td>48.4.17</td>
<td>DMA requests</td>
<td>2100</td>
</tr>
<tr>
<td>48.4.18</td>
<td>Debug mode</td>
<td>2101</td>
</tr>
</tbody>
</table>

### 48.5 I2C low-power modes
- 2101

### 48.6 I2C interrupts
- 2102

### 48.7 I2C registers

- **48.7.1** I2C control register 1 (I2C_CR1) | 2103
- **48.7.2** I2C control register 2 (I2C_CR2) | 2106
- **48.7.3** I2C own address 1 register (I2C_OAR1) | 2108
- **48.7.4** I2C own address 2 register (I2C_OAR2) | 2109
- **48.7.5** I2C timing register (I2C_TIMINGR) | 2110
- **48.7.6** I2C timeout register (I2C_TIMEOUTR) | 2111
- **48.7.7** I2C interrupt and status register (I2C_ISR) | 2112
- **48.7.8** I2C interrupt clear register (I2C_ICR) | 2114
- **48.7.9** I2C PEC register (I2C_PECR) | 2115
- **48.7.10** I2C receive data register (I2C_RXDR) | 2116
- **48.7.11** I2C transmit data register (I2C_TXDR) | 2116
- **48.7.12** I2C register map | 2117

## 49 Improved inter-integrated circuit (I3C)

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>49.1</td>
<td>Introduction</td>
<td>2119</td>
</tr>
<tr>
<td>49.2</td>
<td>I3C main features</td>
<td>2119</td>
</tr>
<tr>
<td>49.3</td>
<td>I3C implementation</td>
<td>2121</td>
</tr>
</tbody>
</table>
- **49.3.1** I3C instantiation | 2121
- **49.3.2** I3C wake-up from low-power mode(s) | 2121
- **49.3.3** I3C FIFOs | 2121
- **49.3.4** I3C triggers | 2121
- **49.3.5** I3C interrupt(s) | 2121
- **49.3.6** I3C MIPI® support | 2122
49.4 I3C block diagram

49.5 I3C pins and internal signals

49.6 I3C reset and clocks
   49.6.1 I3C reset
   49.6.2 I3C clocks and requirements

49.7 I3C peripheral state and programming
   49.7.1 I3C peripheral state
   49.7.2 I3C controller state and programming sequence
   49.7.3 I3C target state and programming sequence

49.8 I3C registers and programming
   49.8.1 I3C register set, as controller/target
   49.8.2 I3C registers and fields use vs. peripheral state, as controller
   49.8.3 I3C registers and fields usage vs. peripheral state, as target

49.9 I3C bus transfers and programming
   49.9.1 I3C command set (CCCs), as controller/target
   49.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller
   49.9.3 I3C broadcast ENTDAA CCC transfer, as controller
   49.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller
   49.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target
   49.9.6 I3C broadcast ENTDAA CCC transfer, as target
   49.9.7 I3C broadcast DEFTGTS CCC transfer, as target
   49.9.8 I3C broadcast DEFGRPA CCC transfer, as target
   49.9.9 I3C direct GETSTATUS CCC response, as target
   49.9.10 I3C private read/write transfer, as controller
   49.9.11 I3C private read/write transfer, as target
   49.9.12 Legacy I2C read/write transfer, as controller
   49.9.13 I3C IBI transfer, as controller/target
   49.9.14 I3C hot-join request transfer, as controller/target
   49.9.15 I3C controller-role request transfer, as controller/target

49.10 I3C FIFOs management, as controller
   49.10.1 C-FIFO management, as controller
   49.10.2 TX-FIFO management, as controller
   49.10.3 RX-FIFO management, as controller
   49.10.4 S-FIFO management, as controller

49.11 I3C FIFOs management, as target
49.11.1 RX-FIFO management, as target ........................................... 2169
49.11.2 TX-FIFO management, as target ........................................... 2170
49.12 I3C error management ............................................................. 2173
49.12.1 Controller error management ................................................... 2173
49.12.2 Target error management ....................................................... 2175
49.13 I3C wake-up from low-power mode(s) ........................................ 2176
49.13.1 Wake-up from Stop ................................................................. 2176
49.14 I3C in low-power modes ........................................................... 2179
49.15 I3C interrupts ........................................................................... 2180
49.16 I3C registers .............................................................................. 2181
49.16.1 I3C message control register (I3C_CR) ....................................... 2181
49.16.2 I3C message control register [alternate] (I3C_CR) ....................... 2183
49.16.3 I3C configuration register (I3C_CFGR) ...................................... 2185
49.16.4 I3C receive data byte register (I3C_RDR) .................................. 2190
49.16.5 I3C receive data word register (I3C_RDWR) .............................. 2190
49.16.6 I3C transmit data byte register (I3C_TDR) ................................. 2191
49.16.7 I3C transmit data word register (I3C_TDWR) .............................. 2192
49.16.8 I3C IBI payload data register (I3C_IBIDR) ................................. 2194
49.16.9 I3C target transmit configuration register (I3C_TGTTRD) ................ 2195
49.16.10 I3C status register (I3C_SR) .................................................... 2196
49.16.11 I3C status error register (I3C_SER) ......................................... 2197
49.16.12 I3C received message register (I3C_RMR) ............................... 2199
49.16.13 I3C event register (I3C_EVR) ................................................ 2200
49.16.14 I3C interrupt enable register (I3C_IER) .................................... 2204
49.16.15 I3C clear event register (I3C_CEVR) ....................................... 2206
49.16.16 I3C own device characteristics register (I3C_DEVR0) .................. 2208
49.16.17 I3C device x characteristics register (I3C_DEVRx) ..................... 2210
49.16.18 I3C maximum read length register (I3C_MAXRLR) ..................... 2212
49.16.19 I3C maximum write length register (I3C_MAXWLR) .................... 2213
49.16.20 I3C timing register 0 (I3C_TIMINGR0) ...................................... 2214
49.16.21 I3C timing register 1 (I3C_TIMINGR1) ...................................... 2215
49.16.22 I3C timing register 2 (I3C_TIMINGR2) ...................................... 2217
49.16.23 I3C bus characteristics register (I3C_BCR) ............................... 2218
49.16.24 I3C device characteristics register (I3C_DCR) ........................... 2219
49.16.25 I3C get capability register (I3C_GETCAPR) .............................. 2220
49.16.26 I3C controller-role capability register (I3C_CRCAPR) .................. 2221
49.16.27 I3C get max data speed register (I3C_GETMXDSR) .......................... 2222
49.16.28 I3C extended provisioned ID register (I3C_EPIDR) ....................... 2224
49.16.29 I3C register map ........................................................................... 2224

50  Universal synchronous/asynchronous receiver transmitter (USART/UART) .................................................. 2228

50.1  Introduction ......................................................................................... 2228
50.2  USART main features ....................................................................... 2228
50.3  USART extended features .................................................................. 2229
50.4  USART implementation .................................................................... 2229
50.5  USART functional description .......................................................... 2231
  50.5.1  USART block diagram ................................................................. 2231
  50.5.2  USART pins and internal signals ............................................... 2231
  50.5.3  USART clocks ............................................................................. 2233
  50.5.4  USART character description ................................................... 2233
  50.5.5  USART FIFOs and thresholds .................................................... 2236
  50.5.6  USART transmitter ..................................................................... 2236
  50.5.7  USART receiver .......................................................................... 2239
  50.5.8  USART baud rate generation ...................................................... 2246
  50.5.9  Tolerance of the USART receiver to clock deviation ................. 2248
  50.5.10  USART auto baud rate detection .............................................. 2249
  50.5.11  USART multiprocessor communication .................................... 2251
  50.5.12  USART Modbus communication ............................................... 2253
  50.5.13  USART parity control ............................................................... 2254
  50.5.14  USART LIN (local interconnection network) mode .................... 2255
  50.5.15  USART synchronous mode ....................................................... 2257
  50.5.16  USART single-wire Half-duplex communication ....................... 2261
  50.5.17  USART receiver timeout .......................................................... 2261
  50.5.18  USART Smartcard mode ........................................................... 2262
  50.5.19  USART IrDA SIR ENDEC block ............................................... 2266
  50.5.20  Continuous communication using USART and DMA ............... 2269
  50.5.21  RS232 Hardware flow control and RS485 Driver Enable .......... 2271
  50.5.22  USART low-power management .............................................. 2274
50.6  USART in low-power modes ............................................................ 2277
50.7  USART interrupts ............................................................................ 2277
50.8  USART registers ............................................................................... 2280
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.8.1</td>
<td>USART control register 1 (USART_CR1)</td>
</tr>
<tr>
<td>50.8.2</td>
<td>USART control register 1 [alternate] (USART_CR1)</td>
</tr>
<tr>
<td>50.8.3</td>
<td>USART control register 2 (USART_CR2)</td>
</tr>
<tr>
<td>50.8.4</td>
<td>USART control register 3 (USART_CR3)</td>
</tr>
<tr>
<td>50.8.5</td>
<td>USART baud rate register (USART_BRR)</td>
</tr>
<tr>
<td>50.8.6</td>
<td>USART guard time and prescaler register (USART_GTPR)</td>
</tr>
<tr>
<td>50.8.7</td>
<td>USART receiver timeout register (USART_RTOR)</td>
</tr>
<tr>
<td>50.8.8</td>
<td>USART request register (USART_RQR)</td>
</tr>
<tr>
<td>50.8.9</td>
<td>USART interrupt and status register (USART_ISR)</td>
</tr>
<tr>
<td>50.8.10</td>
<td>USART interrupt and status register [alternate] (USART_ISR)</td>
</tr>
<tr>
<td>50.8.11</td>
<td>USART interrupt flag clear register (USART_ICR)</td>
</tr>
<tr>
<td>50.8.12</td>
<td>USART receive data register (USART_RDR)</td>
</tr>
<tr>
<td>50.8.13</td>
<td>USART transmit data register (USART_TDR)</td>
</tr>
<tr>
<td>50.8.14</td>
<td>USART prescaler register (USART_PRESC)</td>
</tr>
<tr>
<td>50.8.15</td>
<td>USART register map</td>
</tr>
</tbody>
</table>

51 Low-power universal asynchronous receiver transmitter (LPUART) .......................... 2316

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>51.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>51.2</td>
<td>LPUART main features</td>
</tr>
<tr>
<td>51.3</td>
<td>LPUART implementation</td>
</tr>
<tr>
<td>51.4</td>
<td>LPUART functional description</td>
</tr>
<tr>
<td>51.4.1</td>
<td>LPUART block diagram</td>
</tr>
<tr>
<td>51.4.2</td>
<td>LPUART pins and internal signals</td>
</tr>
<tr>
<td>51.4.3</td>
<td>LPUART clocks</td>
</tr>
<tr>
<td>51.4.4</td>
<td>LPUART character description</td>
</tr>
<tr>
<td>51.4.5</td>
<td>LPUART FIFOs and thresholds</td>
</tr>
<tr>
<td>51.4.6</td>
<td>LPUART transmitter</td>
</tr>
<tr>
<td>51.4.7</td>
<td>LPUART receiver</td>
</tr>
<tr>
<td>51.4.8</td>
<td>LPUART baud rate generation</td>
</tr>
<tr>
<td>51.4.9</td>
<td>Tolerance of the LPUART receiver to clock deviation</td>
</tr>
<tr>
<td>51.4.10</td>
<td>LPUART multiprocessor communication</td>
</tr>
<tr>
<td>51.4.11</td>
<td>LPUART parity control</td>
</tr>
<tr>
<td>51.4.12</td>
<td>LPUART single-wire Half-duplex communication</td>
</tr>
<tr>
<td>51.4.13</td>
<td>Continuous communication using DMA and LPUART</td>
</tr>
<tr>
<td>51.4.14</td>
<td>RS232 Hardware flow control and RS485 Driver Enable</td>
</tr>
<tr>
<td>51.4.15</td>
<td>LPUART low-power management</td>
</tr>
</tbody>
</table>
51.5 LPUART in low-power modes ................................................. 2345
51.6 LPUART interrupts ..................................................... 2346
51.7 LPUART registers .................................................... 2347
   51.7.1 LPUART control register 1 (LPUART_CR1) ......................... 2347
   51.7.2 LPUART control register 1 [alternate] (LPUART_CR1) ............ 2350
   51.7.3 LPUART control register 2 (LPUART_CR2) ......................... 2353
   51.7.4 LPUART control register 3 (LPUART_CR3) ......................... 2355
   51.7.5 LPUART baud rate register (LPUART_BRR) ....................... 2358
   51.7.6 LPUART request register (LPUART_RQR) ......................... 2359
   51.7.7 LPUART interrupt and status register (LPUART_ISR) ............. 2359
   51.7.8 LPUART interrupt and status register [alternate] (LPUART_ISR) ... 2364
   51.7.9 LPUART interrupt flag clear register (LPUART_ICR) ............ 2367
   51.7.10 LPUART receive data register (LPUART_RDR) ................. 2368
   51.7.11 LPUART transmit data register (LPUART_TDR) ................. 2369
   51.7.12 LPUART prescaler register (LPUART_PRESC) .................... 2369
   51.7.13 LPUART register map ......................................... 2370

52 Serial peripheral interface (SPI) ........................................... 2372
   52.1 Introduction ..................................................... 2372
   52.2 SPI main features ............................................... 2372
   52.3 SPI implementation ............................................. 2373
   52.4 SPI functional description ...................................... 2374
      52.4.1 SPI block diagram ........................................ 2374
      52.4.2 SPI pins and internal signals ............................. 2375
      52.4.3 SPI communication general aspects ....................... 2376
      52.4.4 Communications between one master and one slave .......... 2376
      52.4.5 Standard multislave communication ....................... 2379
      52.4.6 Multimaster communication ................................ 2382
      52.4.7 Slave select (SS) pin management ........................ 2382
      52.4.8 Ready pin (RDY) management .............................. 2386
      52.4.9 Communication formats .................................... 2386
      52.4.10 Configuring the SPI ....................................... 2388
      52.4.11 Enabling the SPI ......................................... 2389
      52.4.12 SPI data transmission and reception procedures .......... 2389
      52.4.13 Disabling the SPI ....................................... 2393
      52.4.14 Data packing ............................................ 2395
52.4.15 Communication using DMA (direct memory addressing) ............... 2396
52.5 SPI specific modes and control ........................................... 2398
  52.5.1 TI mode ................................................................. 2398
  52.5.2 SPI error flags ....................................................... 2399
  52.5.3 CRC computation ..................................................... 2401
52.6 SPI low-power modes ....................................................... 2403
52.7 SPI interrupts ............................................................. 2403
52.8 I2S main features .......................................................... 2404
52.9 I2S functional description ................................................ 2404
  52.9.1 I2S general description ............................................... 2404
  52.9.2 Pin sharing with SPI function ....................................... 2405
  52.9.3 Bitfields usable in I2S/PCM mode .................................. 2405
  52.9.4 Slave and master modes .............................................. 2406
  52.9.5 Supported audio protocols .......................................... 2406
  52.9.6 Additional serial interface flexibility .............................. 2412
  52.9.7 Startup sequence ..................................................... 2414
  52.9.8 Stop sequence ........................................................ 2416
  52.9.9 Clock generator ...................................................... 2417
  52.9.10 Internal FIFOs ......................................................... 2419
  52.9.11 FIFOs status flags ................................................. 2420
  52.9.12 Handling of underrun situation ................................... 2420
  52.9.13 Handling of overrun situation ..................................... 2421
  52.9.14 Frame error detection .............................................. 2422
  52.9.15 DMA interface ....................................................... 2424
  52.9.16 Programming examples ............................................. 2424
52.10 I2S interrupts .............................................................. 2427
52.11 SPI/I2S registers .......................................................... 2427
  52.11.1 SPI/I2S control register 1 (SPI_CR1) ............................... 2427
  52.11.2 SPI/I2S control register 2 (SPI_CR2) ............................... 2429
  52.11.3 SPI/I2S configuration register 1 (SPI_CFG1) ..................... 2430
  52.11.4 SPI/I2S configuration register 2 (SPI_CFG2) ..................... 2433
  52.11.5 SPI/I2S interrupt enable register (SPI_IER) .................... 2436
  52.11.6 SPI/I2S status register (SPI_SR) ................................ 2437
  52.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR) .... 2439
  52.11.8 SPI/I2S transmit data register (SPI_TXDR) ..................... 2440
  52.11.9 SPI/I2S receive data register (SPI_RXDR) ...................... 2441
52.11.10 SPI/I2S polynomial register (SPI_CRCPOLY) ........................................ 2441
52.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC) .................................... 2442
52.11.12 SPI/I2S receiver CRC register (SPI_RXCRC) ........................................ 2443
52.11.13 SPI/I2S underrun data register (SPI_UDRDR) ....................................... 2443
52.11.14 SPI/I2S configuration register (SPI_I2SCFGR) ...................................... 2444
52.11.15 SPI/I2S register map ................................................................. 2446

53 Serial audio interface (SAI) ............................................................. 2448
53.1 Introduction .................................................................................. 2448
53.2 SAI main features .......................................................................... 2448
53.3 SAI implementation ....................................................................... 2449
53.4 SAI functional description ............................................................ 2449
  53.4.1 SAI block diagram ........................................................................ 2449
  53.4.2 SAI pins and internal signals ........................................................ 2451
  53.4.3 Main SAI modes ........................................................................... 2451
  53.4.4 SAI synchronization mode ............................................................. 2452
  53.4.5 Audio data size ............................................................................ 2453
  53.4.6 Frame synchronization ................................................................ 2454
  53.4.7 Slot configuration ........................................................................ 2457
  53.4.8 SAI clock generator ...................................................................... 2459
  53.4.9 Internal FIFOs ............................................................................. 2462
  53.4.10 PDM interface ............................................................................ 2464
  53.4.11 AC’97 link controller ................................................................. 2472
  53.4.12 SPDIF output ............................................................................. 2474
  53.4.13 Specific features .......................................................................... 2477
  53.4.14 Error flags ................................................................................ 2481
  53.4.15 Disabling the SAI ....................................................................... 2484
  53.4.16 SAI DMA interface ................................................................. 2484
53.5 SAI interrupts ................................................................................ 2485
53.6 SAI registers .................................................................................. 2486
  53.6.1 SAI global configuration register (SAI_GCR) .................................... 2486
  53.6.2 SAI configuration register 1 (SAI_ACR1) ........................................ 2487
  53.6.3 SAI configuration register 1 (SAI_BCR1) ........................................ 2489
  53.6.4 SAI configuration register 2 (SAI_ACR2) ........................................ 2492
  53.6.5 SAI configuration register 2 (SAI_BCR2) ........................................ 2494
  53.6.6 SAI frame configuration register (SAI_AFRCCR) ......................... 2496
53.6.7 SAI frame configuration register (SAI_BFRCR) .............................................. 2498
53.6.8 SAI slot register (SAI_ASLOTR) ................................................................. 2499
53.6.9 SAI slot register (SAI_BSLOTR) ................................................................. 2500
53.6.10 SAI interrupt mask register (SAI_AIM) ....................................................... 2501
53.6.11 SAI interrupt mask register (SAI_BIM) ....................................................... 2503
53.6.12 SAI status register (SAI_ASR) ................................................................. 2504
53.6.13 SAI status register (SAI_BSR) ................................................................. 2506
53.6.14 SAI clear flag register (SAI_ACLRFR) ....................................................... 2508
53.6.15 SAI clear flag register (SAI_BCLRFR) ....................................................... 2509
53.6.16 SAI data register (SAI_ADR) ................................................................. 2510
53.6.17 SAI data register (SAI_BDR) ................................................................. 2511
53.6.18 SAI PDM control register (SAI_PDMCR) .................................................. 2511
53.6.19 SAI PDM delay register (SAI_PDMDL) ...................................................... 2512
53.6.20 SAI register map ................................................................. 2515

54 FD controller area network (FDCAN) ......................................................... 2517

54.1 Introduction ................................................................. 2517
54.2 FDCAN main features ................................................................. 2519
54.3 FDCAN functional description ................................................................. 2520
  54.3.1 Bit timing ................................................................. 2521
  54.3.2 Operating modes ................................................................. 2522
  54.3.3 Message RAM ................................................................. 2531
  54.3.4 FIFO acknowledge handling ................................................................. 2539
  54.3.5 FDCAN Rx FIFO element ................................................................. 2540
  54.3.6 FDCAN Tx buffer element ................................................................. 2542
  54.3.7 FDCAN Tx event FIFO element ................................................................. 2544
  54.3.8 FDCAN Standard message ID filter element ................................................ 2545
  54.3.9 FDCAN Extended message ID filter element ................................................ 2546
54.4 FDCAN registers ................................................................. 2547
  54.4.1 FDCAN core release register (FDCAN_CREL) .............................................. 2547
  54.4.2 FDCAN endian register (FDCAN_ENDN) .................................................. 2547
  54.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP) .................. 2548
  54.4.4 FDCAN test register (FDCAN_TEST) .................................................. 2549
  54.4.5 FDCAN RAM watchdog register (FDCAN_RWD) .............................................. 2550
  54.4.6 FDCAN CC control register (FDCAN_CCCR) .............................................. 2550
  54.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP) ........... 2552
  54.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC) ........... 2553
54.4.9 FDCAN timestamp counter value register (FDCAN_TSCV) 2554
54.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC) 2554
54.4.11 FDCAN timeout counter value register (FDCAN_TOCV) 2555
54.4.12 FDCAN error counter register (FDCAN_ECR) 2555
54.4.13 FDCAN protocol status register (FDCAN_PSR) 2556
54.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR) 2558
54.4.15 FDCAN interrupt register (FDCAN_IR) 2559
54.4.16 FDCAN interrupt enable register (FDCAN_IE) 2561
54.4.17 FDCAN interrupt line select register (FDCAN_ILS) 2563
54.4.18 FDCAN interrupt line enable register (FDCAN_ILE) 2564
54.4.19 FDCAN global filter configuration register (FDCAN_RXGFC) 2565
54.4.20 FDCAN extended ID and mask register (FDCAN_XIDAM) 2566
54.4.21 FDCAN high-priority message status register (FDCAN_HPMS) 2567
54.4.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S) 2567
54.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A) 2568
54.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S) 2568
54.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A) 2569
54.4.26 FDCAN Tx buffer configuration register (FDCAN_TXBC) 2570
54.4.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS) 2570
54.4.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP) 2571
54.4.29 FDCAN Tx buffer add request register (FDCAN_TXBAR) 2572
54.4.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR) 2572
54.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO) 2573
54.4.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF) 2573
54.4.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE) 2574
54.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE) 2574
54.4.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS) 2575
54.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA) 2575
54.4.37 FDCAN CFG clock divider register (FDCAN_CKDIV) 2576
54.4.38 FDCAN register map 2576

55 Universal serial bus full-speed host/device interface (USB) 2580

55.1 Introduction 2580
55.2 USB main features 2580
55.3 USB implementation 2580
55.4 USB functional description ................................. 2581
  55.4.1 Description of USB blocks used in both Device and Host modes 2583
  55.4.2 Description of host frame scheduler (HFS) specific to Host mode 2584
55.5 Programming considerations for Device and Host modes .......... 2585
  55.5.1 Generic USB Device programming .......................... 2585
  55.5.2 System and power-on reset ............................ 2585
  55.5.3 Double-buffered endpoints and usage in Device mode .......... 2592
  55.5.4 Double buffered channels: usage in Host mode .............. 2594
  55.5.5 Isochronous transfers in Device mode .................... 2595
  55.5.6 Isochronous transfers in Host mode ..................... 2596
  55.5.7 Suspend/resume events .................................. 2597
55.6 USB and USB SRAM registers .................................. 2601
  55.6.1 Common registers .................................. 2601
  55.6.2 Buffer descriptor table ................................ 2620
  55.6.3 USB register map .................................... 2624
56 USB Type-C®/USB Power Delivery interface (UCPD) ............... 2626
  56.1 Introduction ......................................... 2626
  56.2 UCPD main features .................................. 2626
  56.3 UCPD implementation .................................. 2626
  56.4 UCPD functional description ............................. 2627
  56.4.1 UCPD block diagram ................................ 2628
  56.4.2 UCPD reset and clocks ............................... 2629
  56.4.3 Physical layer protocol .............................. 2630
  56.4.4 UCPD BMC transmitter ................................ 2636
  56.4.5 UCPD BMC receiver .................................. 2638
  56.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd) .......... 2639
  56.4.7 UCPD Type-C voltage monitoring and de-bouncing .......... 2640
  56.4.8 UCPD fast role swap (FRS) ............................ 2640
  56.4.9 UCPD DMA Interface .................................. 2640
  56.4.10 Wake-up from Stop mode .............................. 2640
  56.5 UCPD programming sequences .............................. 2641
    56.5.1 Initialization phase ................................ 2641
    56.5.2 Type-C state machine handling ......................... 2641
    56.5.3 USB PD transmit ................................... 2643
    56.5.4 USB PD receive ................................... 2644
56.5.5 UCPD software trimming ........................................ 2645
56.6 UCPD low-power modes ........................................... 2645
56.7 UCPD interrupts ................................................... 2646
56.8 UCPD registers ..................................................... 2647
  56.8.1 UCPD configuration register 1 (UCPD_CFG1) ................. 2647
  56.8.2 UCPD configuration register 2 (UCPD_CFG2) ................. 2649
  56.8.3 UCPD configuration register 3 (UCPD_CFG3) ................. 2650
  56.8.4 UCPD control register (UCPD_CR) .......................... 2650
  56.8.5 UCPD interrupt mask register (UCPD_IMR) ................... 2653
  56.8.6 UCPD status register (UCPD_SR) ........................... 2654
  56.8.7 UCPD interrupt clear register (UCPD_ICR) ................. 2657
  56.8.8 UCPD Tx ordered set type register (UCPD_TX_ORDSETR) .... 2658
  56.8.9 UCPD Tx payload size register (UCPD_TX_PAYSZR) ....... 2659
  56.8.10 UCPD Tx data register (UCPD_TXDR) ...................... 2659
  56.8.11 UCPD Rx ordered set register (UCPD_RX_ORDSETR) ....... 2660
  56.8.12 UCPD Rx payload size register (UCPD_RX_PAYSZR) ....... 2661
  56.8.13 UCPD receive data register (UCPD_RXDR) ................. 2661
  56.8.14 UCPD Rx ordered set extension register 1 
    (UCPD_RX_ORDEXTR1) ........................................... 2662
  56.8.15 UCPD Rx ordered set extension register 2 
    (UCPD_RX_ORDEXTR2) ........................................... 2662
  56.8.16 UCPD register map ........................................ 2663

57 Ethernet (ETH): media access control 
(MAC) with DMA controller ........................................ 2665
  57.1 Ethernet introduction ........................................ 2665
  57.2 Ethernet main features ....................................... 2665
    57.2.1 Standard compliance .................................... 2665
    57.2.2 MAC features .......................................... 2665
    57.2.3 Transaction layer (MTL) features ....................... 2667
    57.2.4 DMA block features .................................... 2668
    57.2.5 Bus interface features ................................ 2668
  57.3 Ethernet pins and internal signals .......................... 2669
  57.4 Ethernet architecture ....................................... 2670
    57.4.1 DMA controller ........................................ 2671
    57.4.2 MTL ................................................... 2680
    57.4.3 MAC ................................................... 2680
57.5 Ethernet functional description: MAC
  57.5.1 Double VLAN processing
  57.5.2 Source address and VLAN insertion, replacement, or deletion
  57.5.3 Packet filtering
  57.5.4 IEEE 1588 timestamp support
  57.5.5 Checksum offload engine
  57.5.6 TCP segmentation offload
  57.5.7 IPv4 ARP offload
  57.5.8 Loopback
  57.5.9 Flow control
  57.5.10 MAC management counters
  57.5.11 Interrupts generated by the MAC
  57.5.12 MAC and MMC register descriptions

57.6 Ethernet functional description: PHY interfaces
  57.6.1 Station management agent (SMA)
  57.6.2 Media independent interface (MII)
  57.6.3 Reduced media independent interface (RMII)

57.7 Ethernet low-power modes
  57.7.1 Low-power management
  57.7.2 Energy Efficient Ethernet (EEE)

57.8 Ethernet interrupts
  57.8.1 DMA interrupts
  57.8.2 MTL interrupts
  57.8.3 MAC Interrupts

57.9 Ethernet programming model
  57.9.1 DMA initialization
  57.9.2 MTL initialization
  57.9.3 MAC initialization
  57.9.4 Performing normal receive and transmit operation
  57.9.5 Stopping and starting transmission
  57.9.6 Programming guidelines for switching to new descriptor list in RxDMA
  57.9.7 Programming guidelines for switching the AHB clock frequency
  57.9.8 Programming guidelines for MII link state transitions
  57.9.9 Programming guidelines for IEEE 1588 timestamping
  57.9.10 Programming guidelines for PTP offload feature
  57.9.11 Programming guidelines for Energy Efficient Ethernet (EEE)
57.9.12 Programming guidelines for flexible pulse-per-second (PPS) output .......................... 2776
57.9.13 Programming guidelines for TSO ................................................................. 2778
57.9.14 Programming guidelines to perform VLAN filtering on the receive .................. 2779

57.10 Descriptors .................................................. 2779
57.10.1 Descriptor overview ................................................................. 2779
57.10.2 Descriptor structure ................................................................. 2780
57.10.3 Transmit descriptor ........................................................................ 2782
57.10.4 Receive descriptor ........................................................................... 2795

57.11 Ethernet registers .............................................. 2807
57.11.1 Ethernet register maps ................................................................. 2807
57.11.2 Ethernet DMA registers .................................................................... 2807
57.11.3 Ethernet MTL registers ...................................................................... 2833
57.11.4 Ethernet MAC and MMC registers ................................................. 2845

58 Debug support (DBG) ............................................... 2942
58.1 Introduction ............................................................... 2942
58.2 DBG functional description ............................................................ 2943
58.2.1 DBG block diagram ................................................................. 2943
58.2.2 DBG pins and internal signals ..................................................... 2943
58.2.3 DBG reset and clocks ................................................................. 2944
58.2.4 DBG power domains ................................................................. 2944
58.2.5 Debug and low-power modes ....................................................... 2944
58.2.6 Security ................................................................................... 2945
58.2.7 Debug authentication .................................................................... 2947
58.3 Serial-wire and JTAG debug port (SWJ-DP) ........................................ 2949
58.3.1 JTAG debug port ....................................................................... 2950
58.3.2 Serial-wire debug port ................................................................. 2952
58.3.3 Debug port registers ..................................................................... 2954
58.3.4 Debug port register map and reset values ....................................... 2960
58.4 Access ports .............................................................. 2961
58.4.1 Access port registers .................................................................... 2962
58.4.2 Access port register map ............................................................... 2968
58.5 ROM tables ................................................................. 2969
58.5.1 System ROM table registers ......................................................... 2973
58.5.2 System ROM table register map .................................................... 2977
58.5.3 MCU ROM table registers ............................................................ 2978
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>58.5.4</td>
<td>MCU ROM table register map</td>
<td>2983</td>
</tr>
<tr>
<td>58.5.5</td>
<td>Processor ROM table registers</td>
<td>2984</td>
</tr>
<tr>
<td>58.5.6</td>
<td>Processor ROM table register map</td>
<td>2988</td>
</tr>
<tr>
<td>58.6</td>
<td>Data watchpoint and trace unit (DWT)</td>
<td>2989</td>
</tr>
<tr>
<td>58.6.1</td>
<td>DWT registers</td>
<td>2990</td>
</tr>
<tr>
<td>58.6.2</td>
<td>DWT register map</td>
<td>3004</td>
</tr>
<tr>
<td>58.7</td>
<td>Instrumentation trace macrocell (ITM)</td>
<td>3007</td>
</tr>
<tr>
<td>58.7.1</td>
<td>ITM registers</td>
<td>3008</td>
</tr>
<tr>
<td>58.7.2</td>
<td>ITM register map</td>
<td>3016</td>
</tr>
<tr>
<td>58.8</td>
<td>Breakpoint unit (BPU)</td>
<td>3018</td>
</tr>
<tr>
<td>58.8.1</td>
<td>BPU registers</td>
<td>3018</td>
</tr>
<tr>
<td>58.8.2</td>
<td>BPU register map</td>
<td>3024</td>
</tr>
<tr>
<td>58.9</td>
<td>Embedded Trace Macrocell (ETM)</td>
<td>3025</td>
</tr>
<tr>
<td>58.9.1</td>
<td>ETM registers</td>
<td>3026</td>
</tr>
<tr>
<td>58.9.2</td>
<td>ETM register map</td>
<td>3051</td>
</tr>
<tr>
<td>58.10</td>
<td>Trace port interface unit (TPIU)</td>
<td>3055</td>
</tr>
<tr>
<td>58.10.1</td>
<td>TPIU registers</td>
<td>3056</td>
</tr>
<tr>
<td>58.10.2</td>
<td>TPIU register map</td>
<td>3066</td>
</tr>
<tr>
<td>58.11</td>
<td>Cross-trigger interface (CTI)</td>
<td>3068</td>
</tr>
<tr>
<td>58.11.1</td>
<td>CTI registers</td>
<td>3069</td>
</tr>
<tr>
<td>58.11.2</td>
<td>CTI register map</td>
<td>3080</td>
</tr>
<tr>
<td>58.12</td>
<td>Microcontroller debug unit (DBGMCU)</td>
<td>3082</td>
</tr>
<tr>
<td>58.12.1</td>
<td>Device ID</td>
<td>3082</td>
</tr>
<tr>
<td>58.12.2</td>
<td>Low-power mode emulation</td>
<td>3082</td>
</tr>
<tr>
<td>58.12.3</td>
<td>Peripheral clock freeze</td>
<td>3083</td>
</tr>
<tr>
<td>58.12.4</td>
<td>DBGMCU registers</td>
<td>3085</td>
</tr>
<tr>
<td>58.12.5</td>
<td>DBGMCU register map</td>
<td>3099</td>
</tr>
<tr>
<td>58.13</td>
<td>References</td>
<td>3101</td>
</tr>
<tr>
<td>59</td>
<td>Device electronic signature</td>
<td>3102</td>
</tr>
<tr>
<td>59.1</td>
<td>Unique device ID register (96 bits)</td>
<td>3102</td>
</tr>
<tr>
<td>59.2</td>
<td>Flash size data register</td>
<td>3103</td>
</tr>
<tr>
<td>59.3</td>
<td>Package data register</td>
<td>3104</td>
</tr>
<tr>
<td>60</td>
<td>Important security notice</td>
<td>3105</td>
</tr>
</tbody>
</table>
61       Revision history  ......................................................... 3106
Table 1. Securable peripherals by TZSC ......................................................... 107
Table 2. TrustZone-aware peripherals ......................................................... 110
Table 3. Memory map and peripheral register boundary addresses ................. 113
Table 4. Configuring security attributes with IDAU and SAU ......................... 127
Table 5. MPCW Mx resources ......................................................................... 129
Table 6. MPCBBx resources ........................................................................... 129
Table 7. DMA channel use (security) ............................................................... 132
Table 8. Secure alternate function between peripherals and allocated I/Os .......... 135
Table 9. Non-secure peripheral functions that cannot be connected to secure I/Os . 135
Table 10. Non-secure peripheral functions that can be connected to secure I/Os .... 136
Table 11. TrustZone-aware DBGMCU non-secure accesses management .......... 137
Table 12. DMA channel use (privilege) ......................................................... 141
Table 13. Internal tampers in TAMP ............................................................. 145
Table 14. Effect of low-power modes on TAMP ............................................. 146
Table 15. Accelerated cryptographic operations ............................................. 149
Table 16. Main product life-cycle transitions ................................................ 151
Table 17. Typical product life-cycle phases .................................................. 152
Table 18. Software intellectual property protection with PRODUCT_STAT ....... 157
Table 19. STM32H562/H563 Boot mode when TrustZone is disabled (TZEN = 0xC3) . 161
Table 20. STM32H562/H563 Boot mode when TrustZone is enabled (TZEN = 0xB4) . 161
Table 21. STM32H573x Boot mode when TrustZone is disabled (TZEN = 0xC3) ... 162
Table 22. STM32H573x Boot mode when TrustZone is enabled (TZEN = 0xB4) ... 162
Table 23. GTZC features ................................................................................. 165
Table 24. GTZC1 sub-block address offset .................................................... 166
Table 25. MPCW M resource assignment ........................................................ 166
Table 26. MPCBB resource assignment ........................................................ 166
Table 27. Secure properties of subregions A and B ....................................... 169
Table 28. Privileged properties of subregions A and B .................................. 169
Table 29. GTZC interrupt request ................................................................. 171
Table 30. GTZC1 TZSC register map and reset values .................................... 189
Table 31. GTZC1 TZIC register map and reset values ...................................... 217
Table 32. GTZC1 MPCBBz register map and reset values (z = 1 to 3) .............. 221
Table 33. Internal SRAMs features ............................................................... 223
Table 34. Effect of low-power modes on RAMCFG ..................................... 225
Table 35. RAMCFG interrupt requests ......................................................... 225
Table 36. RAMCFG register map and reset values ....................................... 231
Table 37. FLASH recommended number of wait states and programming delay .. 240
Table 38. Flash memory OTP organization .................................................. 249
Table 39. Read-only public data organization .............................................. 250
Table 40. Memory map and swapping option ............................................... 252
Table 41. Recommended reactions to FLASH_OPSR contents ....................... 254
Table 42. Option-byte organization .............................................................. 257
Table 43. Specific OB modifying rules .......................................................... 261
Table 44. OB modifiable in closed product .................................................... 261
Table 45. Option-byte key area ...................................................................... 263
Table 46. Default secure watermark .............................................................. 270
Table 47. Flash memory TZ protection summary ......................................... 270
Table 48. TZ protection and bank or mass erase summary ............................ 271
Table 49. Secure watermark-based area .................................................. 272
Table 50. Secure hide protection ........................................................... 273
Table 51. HDP protections summary ....................................................... 274
Table 52. Secure configuration block-based registers access conditions ........ 275
Table 53. Privilege protection summary .................................................. 275
Table 54. Privilege and mass or bank erase ............................................ 276
Table 55. Privilege configuration register access conditions (TZ enabled) ...... 276
Table 56. Privilege configuration register access conditions (TZ disabled) ... 276
Table 57. Flash register accesses ........................................................... 277
Table 58. Flash interface register protection summary ............................... 279
Table 59. High-cycle area protection summary: access to data area address range 280
Table 60. HDP protected definition ......................................................... 280
Table 61. Privilege sectors and the data area - access to data area address range 280
Table 62. Product states, debug states and debug policy ........................... 281
Table 63. PRODUCT_STATE transitions .................................................. 283
Table 64. TZ OBK protection summary .................................................... 284
Table 65. OBK protection summary with TZ disabled ................................ 285
Table 66. Access conditions to secure control register ............................... 285
Table 67. Access conditions to non-secure control register ......................... 285
Table 68. OTP/RO access constraints ....................................................... 286
Table 69. Macros for RSS services ......................................................... 286
Table 70. RSS lib interface functions ....................................................... 287
Table 71. NSS lib interface functions ....................................................... 291
Table 72. Effect of low-power modes on the embedded flash memory .......... 292
Table 73. Locating ECC failure .............................................................. 300
Table 74. Flash interrupt request ........................................................... 303
Table 75. Register map and reset value table .......................................... 349
Table 76. ICACHE features ................................................................. 356
Table 77. TAG memory dimensioning parameters for n-way set associative operating mode (default) 358
Table 78. TAG memory dimensioning parameters for direct-mapped cache mode 359
Table 79. ICACHE cacheability for AHB transaction ................................ 361
Table 80. Memory configurations .......................................................... 361
Table 81. ICACHE remap region size, base address and remap address .......... 362
Table 82. ICACHE interrupts ............................................................... 366
Table 83. ICACHE register map and reset values ...................................... 370
Table 84. DCACHE features ............................................................... 373
Table 85. TAG memory dimensioning parameters .................................... 376
Table 86. DCACHE cacheability for AHB transaction ............................... 378
Table 87. DCACHE interrupts ............................................................. 383
Table 88. DCACHE register map and reset values ..................................... 389
Table 89. PWR input/output pins .......................................................... 392
Table 90. PWR internal input/output signals ........................................... 392
Table 91. Low-power mode summary ..................................................... 409
Table 92. Functionalities depending on the working mode .......................... 409
Table 93. Sleep mode ........................................................... 414
Table 94. Memory shut-off block selection ............................................. 414
Table 95. Stop mode ................................................................. 416
Table 96. Standby mode ............................................................. 418
Table 97. Power modes output states versus MCU power modes ............... 419
Table 98. PWR security configuration summary .............................. 420
Table 99. PWR interrupt requests ....................................................... 422
Table 100. PWR register map and reset values ............................................................... 438
Table 101. RCC input/output signals connected to package pins or balls ........................... 440
Table 102. Reset source identification (RCC_RSR) .......................................................... 442
Table 103. STOPWUCK and STOPKERWUCK description ............................................. 454
Table 104. HSIKERON and CSIKERON behavior ............................................................ 455
Table 105. Kernel clock distribution overview ................................................................. 457
Table 106. RCC security configuration summary ............................................................. 463
Table 107. Interrupt sources and control ....................................................................... 467
Table 108. RCC register map and reset values ................................................................. 469
Table 109. CRS features ................................................................................................ 549
Table 110. CRS internal input/output signals ................................................................. 550
Table 111. Effect of low-power modes on CRS ............................................................... 554
Table 112. Interrupt control bits ..................................................................................... 554
Table 113. CRS register map and reset values ................................................................. 559
Table 114. Port bit configuration ..................................................................................... 562
Table 115. GPIO secured bits ........................................................................................ 570
Table 116. GPIO register map and reset values ............................................................... 579
Table 117. SBS internal input/output signals ................................................................. 583
Table 118. HDPL encoded values .................................................................................. 587
Table 119. SBS boot logic ............................................................................................. 587
Table 120. OBK-HDPL logic ......................................................................................... 592
Table 121. SBS register map and reset values ................................................................. 606
Table 122. Peripherals interconnect matrix .................................................................... 609
Table 123. GPDMA1/2 channel implementation ............................................................ 621
Table 124. GPDMA1/2 autonomous mode and wakeup in low-power modes ..................... 621
Table 125. Programmed GPDMA1/2 request ................................................................. 621
Table 126. Programmed GPDMA1/2 request as a block request ..................................... 626
Table 127. GPDMA1/2 channel with peripheral early termination ................................... 626
Table 128. Programmed GPDMA1/2 request with peripheral early termination .............. 626
Table 129. Programmed GPDMA1/2 trigger ................................................................. 626
Table 130. Programmed GPDMA source/destination burst ........................................... 648
Table 131. Programmed data handling ........................................................................... 653
Table 132. Effect of low-power modes on GPDMA ........................................................ 667
Table 133. GPDMA interrupt requests ........................................................................... 668
Table 134. GPDMA register map and reset values ........................................................ 698
Table 135. STM32H563/H573 and STM32H562 vector table ........................................... 702
Table 136. EXTI signals .............................................................................................. 708
Table 137. EVG signals .............................................................................................. 709
Table 138. EXTI line connections ................................................................................ 709
Table 139. Masking functionality .................................................................................. 713
Table 140. Register protection overview ....................................................................... 714
Table 141. EXTI register map sections ........................................................................... 716
Table 142. EXTI register map and reset values ............................................................... 740
Table 143. CRC internal input/output signals ............................................................... 744
Table 144. CRC register map and reset values .............................................................. 749
Table 145. CORDIC functions ...................................................................................... 751
Table 146. Cosine parameters ...................................................................................... 751
Table 147. Sine parameters ........................................................................................ 752
Table 148. Phase parameters ....................................................................................... 752
Table 149. Modulus parameters .................................................................................. 753
Table 150. Arctangent parameters .............................................................................. 754
Table 151. Hyperbolic cosine parameters ..................................................................... 754
List of tables

Table 203. OCTOSPI implementation ................................................................. 873
Table 204. OCTOSPI input/output pins ............................................................. 875
Table 205. OCTOSPI internal signals ............................................................... 875
Table 206. Command/address phase description ........................................... 884
Table 207. Address alignment cases ............................................................... 898
Table 208. OCTOSPI interrupt requests ......................................................... 900
Table 209. OCTOSPI register map and reset values ....................................... 922
Table 210. SDMMC features ............................................................................ 925
Table 211. SDMMC operation modes SD and SDIO ........................................ 928
Table 212. SDMMC operation modes eMMC .................................................. 928
Table 213. SDMMC internal input/output signals ........................................... 929
Table 214. SDMMC pins .................................................................................. 930
Table 215. SDMMC Command and data phase selection ................................. 931
Table 216. Command token format ............................................................... 937
Table 217. Short response with CRC token format ...................................... 938
Table 218. Short response without CRC token format .................................. 938
Table 219. Long response with CRC token format ....................................... 938
Table 220. Specific Commands overview ....................................................... 939
Table 221. Command path status flags .......................................................... 940
Table 222. Command path error handling ..................................................... 940
Table 223. Data token format ....................................................................... 948
Table 224. Data path status flags and clear bits .......................................... 948
Table 225. Data path error handling ............................................................. 950
Table 226. Data FIFO access ......................................................................... 951
Table 227. Transmit FIFO status flags .......................................................... 952
Table 228. Receive FIFO status flags ............................................................. 953
Table 229. AHB and SDMMC_CK clock frequency relation ......................... 958
Table 230. SDIO special operation control .................................................... 958
Table 231. 4-bit mode Start, interrupt, and CRC-status Signaling detection .... 962
Table 232. CMD12 use cases ......................................................................... 967
Table 233. SDMMC interrupts ...................................................................... 981
Table 234. Response type and SDMMC_RESPxR registers ......................... 989
Table 235. SDMMC register map .................................................................. 1005
Table 236. DLYB internal input/output signals ............................................ 1009
Table 237. Delay block control ..................................................................... 1009
Table 238. DLYB register map and reset values ........................................... 1012
Table 239. ADC features .............................................................................. 1015
Table 240. ADC input/output pins ................................................................. 1017
Table 241. ADC internal input/output signals .............................................. 1017
Table 242. ADC interconnection .................................................................. 1017
Table 243. Configuring the trigger polarity for regular external triggers ........ 1037
Table 244. Configuring the trigger polarity for injected external triggers ....... 1038
Table 245. TSAR timings depending on resolution ........................................ 1050
Table 246. Offset computation versus data resolution ................................. 1053
Table 247. Analog watchdog channel selection ............................................ 1058
Table 248. Analog watchdog 1 comparison ................................................... 1065
Table 249. Analog watchdog 2 and 3 comparison ....................................... 1066
Table 250. Maximum output results versus N and M (gray cells indicate truncation) .................................................. 1069
Table 251. Oversampler operating modes summary ..................................... 1074
Table 252. ADC interrupts ............................................................................ 1093
Table 253. DELAY bits versus ADC resolution ......................................... 1128
Table 254. ADC global register map .............................................................. 1132
Table 255. ADC register map and reset values for each ADC (offset = 0x000 for master ADC, 0x100 for slave ADC). ................................................. 1132
Table 256. ADC register map and reset values (master and slave ADC common registers). ................................................................. 1134
Table 257. DTS internal input/output signals ............................................................. 1137
Table 258. Sampling time configuration ................................................................. 1140
Table 259. Trigger configuration ...................................................................... 1141
Table 260. Temperature sensor behavior in low-power modes ....................... 1143
Table 261. Interrupt control bits .................................................................. 1144
Table 262. DTS register map and reset values .................................................... 1152
Table 263. DAC features .............................................................................. 1154
Table 264. DAC input/output pins .................................................................. 1156
Table 265. DAC internal input/output signals .................................................. 1156
Table 266. DAC interconnection .................................................................... 1156
Table 267. Data format (case of 12-bit data) ..................................................... 1159
Table 268. HFSEL description ....................................................................... 1160
Table 269. Sample and refresh timings ............................................................ 1165
Table 270. Channel output modes summary .................................................... 1167
Table 271. Effect of low-power modes on DAC ............................................... 1172
Table 272. DAC interrupts ............................................................................ 1173
Table 273. DAC register map and reset values ................................................. 1190
Table 274. VREFBUF typical values ................................................................. 1192
Table 275. VREF buffer modes ..................................................................... 1193
Table 276. VREFBUF register map and reset values ........................................ 1195
Table 277. DCMI input/output pins .................................................................. 1197
Table 278. DCMI internal input/output signals ................................................ 1197
Table 279. Positioning of captured data bytes in 32-bit words (8-bit width) .... 1199
Table 280. Positioning of captured data bytes in 32-bit words (10-bit width) ... 1199
Table 281. Positioning of captured data bytes in 32-bit words (12-bit width) ... 1199
Table 282. Positioning of captured data bytes in 32-bit words (14-bit width) ... 1200
Table 283. Data storage in monochrome progressive video format ............... 1205
Table 284. Data storage in RGB progressive video format ............................. 1206
Table 285. Data storage in YCbCr progressive video format ........................... 1206
Table 286. Data storage in YCbCr progressive video format - Y extraction mode 1207
Table 287. DCMI interrupts ......................................................................... 1207
Table 288. DCMI register map and reset values ............................................... 1217
Table 289. PSSI input/output pins ................................................................. 1221
Table 290. PSSI internal input/output signals ................................................ 1221
Table 291. Positioning of captured data bytes in 32-bit words (8-bit width) .... 1222
Table 292. Positioning of captured data bytes in 32-bit words (16-bit width) ... 1223
Table 293. PSSI interrupt requests ................................................................. 1226
Table 294. PSSI register map and reset values ................................................. 1232
Table 295. RNG internal input/output signals .................................................. 1234
Table 296. RNG interrupt requests ................................................................. 1242
Table 297. RNG configurations ................................................................... 1243
Table 298. RNG register map and reset map .................................................... 1249
Table 299. AES versus SAES features ............................................................. 1251
Table 300. AES internal input/output signals .................................................. 1252
Table 301. AES approved symmetric key functions ....................................... 1252
Table 302. Counter mode initialization vector definition ................................. 1262
Table 303. Initialization of IV registers in GCM mode .................................... 1265
Table 304. GCM last block definition .............................................................. 1265
Table 305. Initialization of IV registers in CCM mode ................................. 1271
Table 306. AES data swapping example .................................................. 1275
Table 307. Key endianness in AES_KEYRx registers (128/256-bit keys) .......... 1277
Table 308. IVI bitfield spread over AES_IVRx registers ........................... 1278
Table 309. AES interrupt requests ......................................................... 1279
Table 310. Processing latency for ECB, CBC and CTR ......................... 1280
Table 311. Processing latency for GCM and CCM (in clock cycles) ....... 1280
Table 312. AES register map and reset values ..................................... 1293
Table 313. AES versus SAES features ................................................. 1298
Table 314. SAES internal input/output signals ..................................... 1299
Table 315. SAES approved symmetric key functions ............................ 1299
Table 316. Counter mode initialization vector definition ..................... 1310
Table 317. Initialization of IV registers in GCM mode ......................... 1312
Table 318. GCM last block definition .................................................. 1312
Table 319. Initialization of IV registers in CCM mode ......................... 1318
Table 320. AES data swapping example ............................................. 1327
Table 321. Key endianness in SAES_KEYRx registers (128/256-bit keys) .... 1329
Table 322. IVI bitfield spread over SAES_IVRx registers .................... 1330
Table 323. SAES interrupt requests .................................................... 1333
Table 324. Processing latency for ECB, CBC and CTR ......................... 1334
Table 325. Processing latency for GCM and CCM (in SAES kernel clock cycles) 1334
Table 326. SAES register map and reset values .................................. 1348
Table 327. HASH internal input/output signals .................................. 1351
Table 328. Information on supported hash algorithms ......................... 1352
Table 329. Hash processor outputs ..................................................... 1355
Table 330. HASH interrupt requests ................................................... 1362
Table 331. Processing time (in clock cycle) ........................................ 1362
Table 332. HASH1 register map and reset values ................................ 1370
Table 333. Internal input/output signals ............................................. 1373
Table 334. PKA integer arithmetic functions list ................................ 1374
Table 335. PKA prime field (Fp) elliptic curve functions list ............. 1375
Table 336. Montgomery parameter computation .................................. 1381
Table 337. Modular addition ............................................................ 1381
Table 338. Modular subtraction .......................................................... 1382
Table 339. Montgomery multiplication ................................................ 1383
Table 340. Modular exponentiation (normal mode) ............................ 1383
Table 341. Modular exponentiation (fast mode) .................................. 1384
Table 342. Modular exponentiation (protected mode) ......................... 1384
Table 343. Modular inversion ............................................................ 1385
Table 344. Modular reduction ............................................................. 1385
Table 345. Arithmetic addition ............................................................ 1386
Table 346. Arithmetic subtraction ....................................................... 1386
Table 347. Arithmetic multiplication ................................................... 1386
Table 348. Arithmetic comparison ...................................................... 1387
Table 349. CRT exponentiation ............................................................ 1387
Table 350. Point on elliptic curve Fp check ........................................ 1388
Table 351. ECC Fp scalar multiplication ............................................. 1388
Table 352. ECDSA sign - Inputs .......................................................... 1390
Table 353. ECDSA sign - Outputs ......................................................... 1390
Table 354. Extended ECDSA sign - Extra outputs .............................. 1391
Table 355. ECDSA verification - Inputs ............................................... 1391
Table 356. ECDSA verification - Outputs ............................................. 1392
Table 357. ECC complete addition .................................................. 1392
Table 358. ECC double base ladder .............................................. 1393
Table 359. ECC projective to affine .............................................. 1394
Table 360. Family of supported curves for ECC operations .......... 1395
Table 361. Modular exponentiation .............................................. 1396
Table 362. ECC scalar multiplication .......................................... 1396
Table 363. ECDSA signature average computation time ............... 1397
Table 364. ECDSA verification average computation times .......... 1397
Table 365. ECC double base ladder average computation times .... 1397
Table 366. ECC projective to affine average computation times ... 1397
Table 367. ECC complete addition average computation times .... 1397
Table 368. ECC projective to affine average computation times ... 1397
Table 369. Montgomery parameters average computation times .... 1398
Table 370. PKA interrupt requests ............................................... 1398
Table 371. PKA register map and reset values ............................. 1403
Table 372. OTFDEC internal input/output signals ......................... 1405
Table 373. OTFDEC interrupt requests ......................................... 1409
Table 374. OTFDEC register map and reset values ......................... 1423
Table 375. TIM input/output pins ................................................ 1429
Table 376. TIM internal input/output signals ............................... 1429
Table 377. Interconnect to the tim_t1 input multiplexer ............... 1431
Table 378. Interconnect to the tim_t2 input multiplexer ............... 1431
Table 379. Interconnect to the tim_t3 input multiplexer ............... 1431
Table 380. Interconnect to the tim_t4 input multiplexer ............... 1431
Table 381. Internal trigger connection ....................................... 1431
Table 382. Interconnect to the tim_etr input multiplexer .......... 1432
Table 383. Interconnect to the tim_t1 input multiplexer .......... 1432
Table 384. Timer break interconnect ......................................... 1432
Table 385. System break interconnect ....................................... 1433
Table 386. CCR and ARR register change dithering pattern ......... 1466
Table 387. CCR register change dithering pattern in center-aligned PWM mode 1467
Table 388. Behavior of timer outputs versus tim_brk/tim_brk2 inputs 1479
Table 389. Break protection disarming conditions ...................... 1481
Table 390. Counting direction versus encoder signals (CC1P = CC2P = 0) 1490
Table 391. Counting direction versus encoder signals and polarity settings 1494
Table 392. DMA request ......................................................... 1516
Table 393. Effect of low-power modes on TIM1/TIM8 ................. 1517
Table 394. Interrupt requests ................................................ 1517
Table 395. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature ................................. 1544
Table 396. TIMx register map and reset values .......................... 1567
Table 397. STM32H563/H573 and STM32H562 general purpose timers 1571
Table 398. TIM input/output pins .............................................. 1573
Table 399. TIM internal input/output signals ........................... 1573
Table 400. Interconnect to the tim_t1 input multiplexer .......... 1574
Table 401. Interconnect to the tim_t2 input multiplexer .......... 1574
Table 402. Interconnect to the tim_t3 input multiplexer .......... 1574
Table 403. Interconnect to the tim_t4 input multiplexer .......... 1575
Table 404. TIMx internal trigger connection .............................. 1575
Table 405. Interconnect to the tim_etr input multiplexer .......... 1575
Table 406. CCR and ARR register change dithering pattern ........ 1607
Table 407. CCR register change dithering pattern in center-aligned PWM mode 1608
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>408</td>
<td>Counting direction versus encoder signals (CC1P = CC2P = 0)</td>
<td>1617</td>
</tr>
<tr>
<td>409</td>
<td>Counting direction versus encoder signals and polarity settings</td>
<td>1622</td>
</tr>
<tr>
<td>410</td>
<td>DMA request</td>
<td>1646</td>
</tr>
<tr>
<td>411</td>
<td>Effect of low-power modes on TIM2/TIM3/TIM4/TIM5</td>
<td>1646</td>
</tr>
<tr>
<td>412</td>
<td>Interrupt requests</td>
<td>1647</td>
</tr>
<tr>
<td>413</td>
<td>Output control bit for standard tim_ocx channels</td>
<td>1666</td>
</tr>
<tr>
<td>414</td>
<td>TIM2/TIM3/TIM4/TIM5 register map and reset values</td>
<td>1684</td>
</tr>
<tr>
<td>415</td>
<td>TIM internal input/output signals</td>
<td>1688</td>
</tr>
<tr>
<td>416</td>
<td>TIMx_ARR register change dithering pattern</td>
<td>1698</td>
</tr>
<tr>
<td>417</td>
<td>DMA request</td>
<td>1699</td>
</tr>
<tr>
<td>418</td>
<td>Effect of low-power modes on TIM6/TIM7</td>
<td>1699</td>
</tr>
<tr>
<td>419</td>
<td>Interrupt request</td>
<td>1699</td>
</tr>
<tr>
<td>420</td>
<td>TIMx register map and reset values</td>
<td>1705</td>
</tr>
<tr>
<td>421</td>
<td>TIM input/output pins</td>
<td>1709</td>
</tr>
<tr>
<td>422</td>
<td>TIM internal input/output signals</td>
<td>1709</td>
</tr>
<tr>
<td>423</td>
<td>Interconnect to the tim_ti1 input multiplexer</td>
<td>1711</td>
</tr>
<tr>
<td>424</td>
<td>Interconnect to the tim_ti2 input multiplexer</td>
<td>1711</td>
</tr>
<tr>
<td>425</td>
<td>TIMx internal trigger connection</td>
<td>1711</td>
</tr>
<tr>
<td>426</td>
<td>CCR and ARR register change dithering pattern</td>
<td>1729</td>
</tr>
<tr>
<td>427</td>
<td>Effect of low-power modes on TIM12/TIM13/TIM14</td>
<td>1739</td>
</tr>
<tr>
<td>428</td>
<td>Interrupt requests</td>
<td>1739</td>
</tr>
<tr>
<td>429</td>
<td>Output control bit for standard tim_ocx channels</td>
<td>1752</td>
</tr>
<tr>
<td>430</td>
<td>TIM12 register map and reset values</td>
<td>1756</td>
</tr>
<tr>
<td>431</td>
<td>Output control bit for standard tim_ocx channels</td>
<td>1765</td>
</tr>
<tr>
<td>432</td>
<td>TIM13/TIM14 register map and reset values</td>
<td>1768</td>
</tr>
<tr>
<td>433</td>
<td>TIM input/output pins</td>
<td>1773</td>
</tr>
<tr>
<td>434</td>
<td>TIM internal input/output signals</td>
<td>1774</td>
</tr>
<tr>
<td>435</td>
<td>Interconnect to the tim_ti1 input multiplexer</td>
<td>1775</td>
</tr>
<tr>
<td>436</td>
<td>Interconnect to the tim_ti2 input multiplexer</td>
<td>1775</td>
</tr>
<tr>
<td>437</td>
<td>TIMx internal trigger connection</td>
<td>1775</td>
</tr>
<tr>
<td>438</td>
<td>Timer break interconnect</td>
<td>1776</td>
</tr>
<tr>
<td>439</td>
<td>System break interconnect</td>
<td>1776</td>
</tr>
<tr>
<td>440</td>
<td>CCR and ARR register change dithering pattern</td>
<td>1796</td>
</tr>
<tr>
<td>441</td>
<td>Break protection disarming conditions</td>
<td>1805</td>
</tr>
<tr>
<td>442</td>
<td>DMA request</td>
<td>1816</td>
</tr>
<tr>
<td>443</td>
<td>Effect of low-power modes on TIM15/TIM16/TIM17</td>
<td>1817</td>
</tr>
<tr>
<td>444</td>
<td>Interrupt requests</td>
<td>1817</td>
</tr>
<tr>
<td>445</td>
<td>Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15)</td>
<td>1833</td>
</tr>
<tr>
<td>446</td>
<td>TIM15 register map and reset values</td>
<td>1846</td>
</tr>
<tr>
<td>447</td>
<td>Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)</td>
<td>1859</td>
</tr>
<tr>
<td>448</td>
<td>TIM16/TIM17 register map and reset values</td>
<td>1873</td>
</tr>
<tr>
<td>449</td>
<td>STM32H563/H573 and STM32H562 LPTIM features</td>
<td>1876</td>
</tr>
<tr>
<td>450</td>
<td>LPTIM1/2/3/5/6 input/output pins</td>
<td>1878</td>
</tr>
<tr>
<td>451</td>
<td>LPTIM4 input/output pins</td>
<td>1878</td>
</tr>
<tr>
<td>452</td>
<td>LPTIM4 input/output pins</td>
<td>1879</td>
</tr>
<tr>
<td>453</td>
<td>LPTIM4 internal signals</td>
<td>1879</td>
</tr>
<tr>
<td>454</td>
<td>LPTIM1/2/3/4/5/6 external trigger connection</td>
<td>1880</td>
</tr>
<tr>
<td>455</td>
<td>LPTIM1/2/3/5/6 input 1 connection</td>
<td>1880</td>
</tr>
<tr>
<td>456</td>
<td>LPTIM1/2/3/5/6 input 2 connection</td>
<td>1880</td>
</tr>
<tr>
<td>457</td>
<td>LPTIM1/2/3/5/6 input capture 1 connection</td>
<td>1880</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>458</td>
<td>LPTIM1 input capture 2 connection</td>
<td>1881</td>
</tr>
<tr>
<td>459</td>
<td>LPTIM2 input capture 2 connection</td>
<td>1881</td>
</tr>
<tr>
<td>460</td>
<td>LPTIM3/5/6 input capture 2 connection</td>
<td>1881</td>
</tr>
<tr>
<td>461</td>
<td>Prescaler division ratios</td>
<td>1882</td>
</tr>
<tr>
<td>462</td>
<td>Encoder counting scenarios</td>
<td>1889</td>
</tr>
<tr>
<td>463</td>
<td>Input capture Glitch filter latency (in counter step unit)</td>
<td>1893</td>
</tr>
<tr>
<td>464</td>
<td>Effect of low-power modes on the LPTIM</td>
<td>1897</td>
</tr>
<tr>
<td>465</td>
<td>Interrupt events</td>
<td>1898</td>
</tr>
<tr>
<td>466</td>
<td>LPTIM register map and reset values</td>
<td>1922</td>
</tr>
<tr>
<td>467</td>
<td>IWDG features</td>
<td>1925</td>
</tr>
<tr>
<td>468</td>
<td>IWDG internal input/output signals</td>
<td>1927</td>
</tr>
<tr>
<td>469</td>
<td>Effect of low-power modes on IWDG</td>
<td>1931</td>
</tr>
<tr>
<td>470</td>
<td>IWDG interrupt request</td>
<td>1933</td>
</tr>
<tr>
<td>471</td>
<td>IWDG register map and reset values</td>
<td>1939</td>
</tr>
<tr>
<td>472</td>
<td>WWDG features</td>
<td>1940</td>
</tr>
<tr>
<td>473</td>
<td>WWDG internal input/output signals</td>
<td>1941</td>
</tr>
<tr>
<td>474</td>
<td>WWDG interrupt requests</td>
<td>1944</td>
</tr>
<tr>
<td>475</td>
<td>WWDG register map and reset values</td>
<td>1946</td>
</tr>
<tr>
<td>476</td>
<td>RTC input/output pins</td>
<td>1950</td>
</tr>
<tr>
<td>477</td>
<td>RTC internal input/output signals</td>
<td>1950</td>
</tr>
<tr>
<td>478</td>
<td>RTC interconnection</td>
<td>1951</td>
</tr>
<tr>
<td>479</td>
<td>RTC pin PC13 configuration</td>
<td>1952</td>
</tr>
<tr>
<td>480</td>
<td>PI8 configuration</td>
<td>1954</td>
</tr>
<tr>
<td>481</td>
<td>RTC_OUT mapping</td>
<td>1955</td>
</tr>
<tr>
<td>482</td>
<td>Effect of low-power modes on RTC</td>
<td>1970</td>
</tr>
<tr>
<td>483</td>
<td>RTC pins functionality over modes</td>
<td>1970</td>
</tr>
<tr>
<td>484</td>
<td>Nonsecure interrupt requests</td>
<td>1971</td>
</tr>
<tr>
<td>485</td>
<td>Secure interrupt requests</td>
<td>1971</td>
</tr>
<tr>
<td>486</td>
<td>RTC register map and reset values</td>
<td>2002</td>
</tr>
<tr>
<td>487</td>
<td>TAMP input/output pins</td>
<td>2008</td>
</tr>
<tr>
<td>488</td>
<td>TAMP internal input/output signals</td>
<td>2008</td>
</tr>
<tr>
<td>489</td>
<td>TAMP interconnection</td>
<td>2009</td>
</tr>
<tr>
<td>490</td>
<td>Device resource x tamper protection</td>
<td>2015</td>
</tr>
<tr>
<td>491</td>
<td>Active tamper output change period</td>
<td>2017</td>
</tr>
<tr>
<td>492</td>
<td>Minimum ATPER value</td>
<td>2018</td>
</tr>
<tr>
<td>493</td>
<td>Active tamper filtered pulse duration</td>
<td>2019</td>
</tr>
<tr>
<td>494</td>
<td>Effect of low-power modes on TAMP</td>
<td>2020</td>
</tr>
<tr>
<td>495</td>
<td>TAMP pins functionality over modes</td>
<td>2020</td>
</tr>
<tr>
<td>496</td>
<td>Interrupt requests</td>
<td>2021</td>
</tr>
<tr>
<td>497</td>
<td>TAMP register map and reset values</td>
<td>2050</td>
</tr>
<tr>
<td>498</td>
<td>STM32H563/H573 and STM32H562 I2C implementation</td>
<td>2053</td>
</tr>
<tr>
<td>499</td>
<td>I2C input/output pins</td>
<td>2055</td>
</tr>
<tr>
<td>500</td>
<td>I2C internal input/output signals</td>
<td>2055</td>
</tr>
<tr>
<td>501</td>
<td>Comparison of analog vs. digital filters</td>
<td>2057</td>
</tr>
<tr>
<td>502</td>
<td>I2C-SMBus specification data setup and hold times</td>
<td>2059</td>
</tr>
<tr>
<td>503</td>
<td>I2C configuration</td>
<td>2064</td>
</tr>
<tr>
<td>504</td>
<td>I2C-SMBus specification clock timings</td>
<td>2075</td>
</tr>
<tr>
<td>505</td>
<td>Examples of timing settings for fI2CCLK = 8 MHz</td>
<td>2085</td>
</tr>
<tr>
<td>506</td>
<td>Examples of timing settings for fI2CCLK = 16 MHz</td>
<td>2085</td>
</tr>
<tr>
<td>507</td>
<td>SMBus timeout specifications</td>
<td>2087</td>
</tr>
<tr>
<td>508</td>
<td>SMBus with PEC configuration</td>
<td>2089</td>
</tr>
<tr>
<td>509</td>
<td>Examples of TIMEOUTA settings (max t_TIMEOUT = 25 ms)</td>
<td>2090</td>
</tr>
</tbody>
</table>
List of tables

Table 510. Examples of TIMEOUTB settings ............................................................... 2090
Table 511. Examples of TIMEOUTA settings (max \( \text{tIDLE} = 50 \mu s \)) .................. 2090
Table 512. Effect of low-power modes on the I2C ........................................... 2101
Table 513. I2C interrupt requests ........................................................................ 2102
Table 514. I2C register map and reset values ...................................................... 2117
Table 515. I3C wake-up ....................................................................................... 2121
Table 516. I3C FIFOs implementation ................................................................. 2121
Table 517. I3C interrupt(s) ................................................................................ 2121
Table 518. I3C peripheral controller/target features versus MIPI v1.1 ............... 2122
Table 519. I3C input/output pins ....................................................................... 2123
Table 520. I3C internal input/output signals ....................................................... 2123
Table 521. I3C register usage ............................................................................ 2135
Table 522. I3C registers/fields usage versus controller state ......................... 2136
Table 523. I3C registers/fields usage versus target state. .............................. 2139
Table 524. List of supported I3C CCCs, as controller/target ............................ 2142
Table 525. I3C controller error management ...................................................... 2173
Table 526. I3C target error management ............................................................. 2175
Table 527. Effect of low-power modes ............................................................... 2179
Table 528. I3C interrupt requests ...................................................................... 2180
Table 529. I3C register map and reset values ..................................................... 2224
Table 530. STM32H563/H573 and STM32H562 features ................................. 2229
Table 531. USART/LPUART features ................................................................. 2230
Table 532. USART input/output pins ................................................................. 2232
Table 533. USART internal input/output signals .............................................. 2233
Table 534. Noise detection from sampled data ................................................ 2245
Table 535. Tolerance of the USART receiver when \( \text{BRR}[3:0] = 0000 \) ............. 2249
Table 536. Tolerance of the USART receiver when \( \text{BRR}[3:0] \) is different from 0000 2249
Table 537. USART frame formats ................................................................... 2254
Table 538. Effect of low-power modes on the USART ................................. 2277
Table 539. USART interrupt requests ............................................................... 2278
Table 540. USART register map and reset values ........................................... 2314
Table 541. STM32H563/H573 and STM32H562 features ................................. 2317
Table 542. USART/LPUART features ................................................................. 2318
Table 543. LPUART input/output pins ............................................................... 2320
Table 544. LPUART internal input/output signals ........................................... 2320
Table 545. Error calculation for programmed baud rates at \( \text{lpuart}_\text{ker}_\text{ck}_\text{pres}\) = 32.768 kHz .................................................. 2331
Table 546. Error calculation for programmed baud rates at \( \text{fCK} = 100 \text{ MHz} \) ........ 2332
Table 547. Tolerance of the LPUART receiver .................................................. 2333
Table 548. Effect of low-power modes on the LPUART ............................... 2345
Table 549. Effect of low-power modes on the USART ................................. 2345
Table 550. LPUART interrupt requests ............................................................... 2346
Table 551. LPUART register map and reset values ........................................... 2370
Table 552. SPI features .................................................................................... 2373
Table 553. SPI/I2S input/output pins ............................................................... 2375
Table 554. SPI internal input/output signals .................................................... 2376
Table 555. Effect of low-power modes on the SPI .......................................... 2403
Table 556. SPI wake-up and interrupt requests ............................................. 2404
Table 557. Bitfields usable in PCM/I2S mode .................................................... 2405
Table 558. WS and CK level before SPI/I2S is enabled when AFCNTR = 1 .... 2414
Table 559. Serial data line swapping .............................................................. 2414
Table 560. CLKGEN programming examples for usual I2S frequencies ........ 2418
Table 561. I2S interrupt requests ................................................................. 2427
Table 562. SPI register map and reset values ................................................. 2446
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>563</td>
<td>STM32H563/H573 and STM32H562 SAI features</td>
<td>2449</td>
</tr>
<tr>
<td>564</td>
<td>SAI internal input/output signals</td>
<td>2451</td>
</tr>
<tr>
<td>565</td>
<td>SAI input/output pins</td>
<td>2451</td>
</tr>
<tr>
<td>566</td>
<td>External synchronization selection</td>
<td>2453</td>
</tr>
<tr>
<td>567</td>
<td>MCLK_x activation conditions</td>
<td>2459</td>
</tr>
<tr>
<td>568</td>
<td>Clock generator programming examples</td>
<td>2462</td>
</tr>
<tr>
<td>569</td>
<td>TDM settings</td>
<td>2469</td>
</tr>
<tr>
<td>570</td>
<td>TDM frame configuration examples</td>
<td>2471</td>
</tr>
<tr>
<td>571</td>
<td>SOPD pattern</td>
<td>2475</td>
</tr>
<tr>
<td>572</td>
<td>Parity bit calculation</td>
<td>2475</td>
</tr>
<tr>
<td>573</td>
<td>Audio sampling frequency versus symbol rates</td>
<td>2476</td>
</tr>
<tr>
<td>574</td>
<td>SAI interrupt sources</td>
<td>2485</td>
</tr>
<tr>
<td>575</td>
<td>SAI register map and reset values</td>
<td>2515</td>
</tr>
<tr>
<td>576</td>
<td>CAN subsystem I/O signals</td>
<td>2517</td>
</tr>
<tr>
<td>577</td>
<td>DLC coding in FDCAN</td>
<td>.</td>
</tr>
<tr>
<td>578</td>
<td>Possible configurations for Frame transmission</td>
<td>2537</td>
</tr>
<tr>
<td>579</td>
<td>Rx FIFO element</td>
<td>2540</td>
</tr>
<tr>
<td>580</td>
<td>Rx FIFO element description</td>
<td>2540</td>
</tr>
<tr>
<td>581</td>
<td>Tx buffer and FIFO element</td>
<td>2542</td>
</tr>
<tr>
<td>582</td>
<td>Tx buffer element description</td>
<td>2542</td>
</tr>
<tr>
<td>583</td>
<td>Tx event FIFO element</td>
<td>2544</td>
</tr>
<tr>
<td>584</td>
<td>Tx event FIFO element description</td>
<td>2544</td>
</tr>
<tr>
<td>585</td>
<td>Standard message ID filter element</td>
<td>2545</td>
</tr>
<tr>
<td>586</td>
<td>Standard message ID filter element field description</td>
<td>2545</td>
</tr>
<tr>
<td>587</td>
<td>Extended message ID filter element</td>
<td>2546</td>
</tr>
<tr>
<td>588</td>
<td>Extended message ID filter element field description</td>
<td>2546</td>
</tr>
<tr>
<td>589</td>
<td>FDCAN register map and reset values</td>
<td>2576</td>
</tr>
<tr>
<td>590</td>
<td>STM32H563/H573 and STM32H562 USB implementation</td>
<td>2580</td>
</tr>
<tr>
<td>591</td>
<td>Double-buffering buffer flag definition</td>
<td>2593</td>
</tr>
<tr>
<td>592</td>
<td>Bulk double-buffering memory buffers usage (Device mode)</td>
<td>2593</td>
</tr>
<tr>
<td>593</td>
<td>Bulk double-buffering memory buffers usage (Host mode)</td>
<td>2595</td>
</tr>
<tr>
<td>594</td>
<td>Isochronous memory buffers usage</td>
<td>2596</td>
</tr>
<tr>
<td>595</td>
<td>Isochronous memory buffers usage</td>
<td>2597</td>
</tr>
<tr>
<td>596</td>
<td>Resume event detection</td>
<td>2599</td>
</tr>
<tr>
<td>597</td>
<td>Resume event detection for host</td>
<td>2600</td>
</tr>
<tr>
<td>598</td>
<td>Reception status encoding</td>
<td>2618</td>
</tr>
<tr>
<td>599</td>
<td>Endpoint/channel type encoding</td>
<td>2618</td>
</tr>
<tr>
<td>600</td>
<td>Endpoint/channel kind meaning</td>
<td>2618</td>
</tr>
<tr>
<td>601</td>
<td>Transmission status encoding</td>
<td>2618</td>
</tr>
<tr>
<td>602</td>
<td>Definition of allocated buffer memory</td>
<td>2621</td>
</tr>
<tr>
<td>603</td>
<td>USB register map and reset values</td>
<td>2624</td>
</tr>
<tr>
<td>604</td>
<td>UCPD implementation</td>
<td>2627</td>
</tr>
<tr>
<td>605</td>
<td>UCPD software trim data</td>
<td>2627</td>
</tr>
<tr>
<td>606</td>
<td>UCPD signals on pins</td>
<td>2628</td>
</tr>
<tr>
<td>607</td>
<td>UCPD internal signals</td>
<td>2629</td>
</tr>
<tr>
<td>608</td>
<td>4b5b symbol encoding table</td>
<td>2631</td>
</tr>
<tr>
<td>609</td>
<td>Ordered sets</td>
<td>2632</td>
</tr>
<tr>
<td>610</td>
<td>Validation of ordered sets</td>
<td>2632</td>
</tr>
<tr>
<td>611</td>
<td>Data size</td>
<td>2633</td>
</tr>
<tr>
<td>612</td>
<td>Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx</td>
<td>2641</td>
</tr>
<tr>
<td>613</td>
<td>Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2)</td>
<td>2643</td>
</tr>
<tr>
<td>614</td>
<td>Effect of low power modes on the UCPD</td>
<td>2645</td>
</tr>
</tbody>
</table>
Table 615. UCPD interrupt requests ................................................................. 2646
Table 616. UCPD register map and reset values ........................................... 2663
Table 617. Ethernet peripheral pins .............................................................. 2669
Table 618. Ethernet internal input/output signals ......................................... 2670
Table 619. Priority scheme for Tx DMA and Rx DMA .................................. 2679
Table 620. Double VLAN processing features in Tx path .............................. 2685
Table 621. Double VLAN processing in Rx path ............................................ 2686
Table 622. VLAN insertion or replacement based on VLTI bit ........................ 2687
Table 623. Destination address filtering ...................................................... 2690
Table 624. Source address filtering .............................................................. 2691
Table 625. VLAN match status ................................................................. 2692
Table 626. Ordinary clock: PTP messages for snapshot ................................. 2695
Table 627. End-to-end transparent clock: PTP messages for snapshot ............. 2696
Table 628. Peer-to-peer transparent clock: PTP messages for snapshot .......... 2697
Table 629. Egress and ingress latency for PHY interfaces ............................. 2700
Table 630. Minimum PTP clock frequency example ..................................... 2701
Table 631. Message format defined in IEEE 1588-2008 ................................. 2702
Table 632. Message format defined in IEEE 1588-2008 ................................. 2702
Table 633. IPv6-UDP PTP packet fields required for control and status ........... 2703
Table 634. Ethernet PTP packet fields required for control and status .......... 2704
Table 635. Timestamp Snapshot Dependency on ETH_MACTSCR bits ............ 2706
Table 636. PTP message generation criteria ............................................... 2712
Table 637. Common PTP message header fields .......................................... 2714
Table 638. MAC Transmit PTP mode and one-step timestamping operation ...... 2717
Table 639. Transmit checksum offload engine functions for different packet types 2722
Table 640. Receive checksum offload engine functions for different packet types 2724
Table 641. TSO: TCP and IP header fields .................................................. 2728
Table 642. Pause packet fields ................................................................. 2733
Table 643. Tx MAC flow control ................................................................. 2734
Table 644. Rx MAC flow control ................................................................. 2734
Table 645. Size of the maximum receive packet ........................................... 2737
Table 646. MCD clock selection ................................................................. 2739
Table 647. MDIO Clause 45 frame structure ................................................ 2740
Table 648. MDIO Clause 22 frame structure ................................................. 2741
Table 649. Remote wake-up packet filter register ....................................... 2752
Table 650. Description of the remote wake-up filter fields .............................. 2753
Table 651. Remote wake-up packet and PMT interrupt generation .................. 2754
Table 652. Transfer complete interrupt behavior ......................................... 2762
Table 653. TDES0 normal descriptor (read format) ....................................... 2782
Table 654. TDES1 normal descriptor (read format) ....................................... 2783
Table 655. TDES2 normal descriptor (read format) ....................................... 2783
Table 656. TDES3 normal descriptor (read format) ....................................... 2784
Table 657. TDES0 normal descriptor (write-back format) ............................... 2787
Table 658. TDES1 normal descriptor (write-back format) ............................... 2787
Table 659. TDES2 normal descriptor (write-back format) ............................... 2788
Table 660. TDES3 normal descriptor (write-back format) ............................... 2788
Table 661. TDES0 context descriptor .......................................................... 2791
Table 662. TDES1 context descriptor .......................................................... 2792
Table 663. TDES2 context descriptor .......................................................... 2792
Table 664. TDES3 context descriptor .......................................................... 2792
Table 665. RDES0 normal descriptor (read format) ....................................... 2796
Table 666. RDES1 normal descriptor (read format) ....................................... 2796
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 667</td>
<td>RDES2 normal descriptor (read format)</td>
<td>2796</td>
</tr>
<tr>
<td>Table 668</td>
<td>RDES3 normal descriptor (read format)</td>
<td>2797</td>
</tr>
<tr>
<td>Table 669</td>
<td>RDES0 normal descriptor (write-back format)</td>
<td>2798</td>
</tr>
<tr>
<td>Table 670</td>
<td>RDES1 normal descriptor (write-back format)</td>
<td>2799</td>
</tr>
<tr>
<td>Table 671</td>
<td>RDES2 normal descriptor (write-back format)</td>
<td>2801</td>
</tr>
<tr>
<td>Table 672</td>
<td>RDES3 normal descriptor (write-back format)</td>
<td>2802</td>
</tr>
<tr>
<td>Table 673</td>
<td>RDES0 context descriptor</td>
<td>2805</td>
</tr>
<tr>
<td>Table 674</td>
<td>RDES1 context descriptor</td>
<td>2806</td>
</tr>
<tr>
<td>Table 675</td>
<td>RDES2 context descriptor</td>
<td>2806</td>
</tr>
<tr>
<td>Table 676</td>
<td>RDES3 context descriptor</td>
<td>2806</td>
</tr>
<tr>
<td>Table 677</td>
<td>ETH_DMA common register map and reset values</td>
<td>2830</td>
</tr>
<tr>
<td>Table 678</td>
<td>ETH_DMA_CH register map and reset values</td>
<td>2830</td>
</tr>
<tr>
<td>Table 679</td>
<td>ETH_MTL register map and reset values</td>
<td>2843</td>
</tr>
<tr>
<td>Table 680</td>
<td>Giant Packet Status based on S2KP and JE Bits</td>
<td>2849</td>
</tr>
<tr>
<td>Table 681</td>
<td>Packet Length based on the CST and ACS bits</td>
<td>2849</td>
</tr>
<tr>
<td>Table 682</td>
<td>Ethernet MAC register map and reset values</td>
<td>2931</td>
</tr>
<tr>
<td>Table 683</td>
<td>SPIRIT address block DBG</td>
<td>2942</td>
</tr>
<tr>
<td>Table 684</td>
<td>JTAG/Serial-wire debug port pins</td>
<td>2943</td>
</tr>
<tr>
<td>Table 685</td>
<td>Trace port pins</td>
<td>2943</td>
</tr>
<tr>
<td>Table 686</td>
<td>Single-wire trace port pins</td>
<td>2944</td>
</tr>
<tr>
<td>Table 687</td>
<td>Authentication signal states with TrustZone enabled (TZEN = 0xB4)</td>
<td>2945</td>
</tr>
<tr>
<td>Table 688</td>
<td>Authentication signal states with TrustZone disabled (TZEN = 0xC3)</td>
<td>2946</td>
</tr>
<tr>
<td>Table 689</td>
<td>Life cycle state and debug states</td>
<td>2946</td>
</tr>
<tr>
<td>Table 690</td>
<td>Definition of data to provision</td>
<td>2949</td>
</tr>
<tr>
<td>Table 691</td>
<td>Permission mask (Endianness: Little Endian)</td>
<td>2949</td>
</tr>
<tr>
<td>Table 692</td>
<td>JTAG-DP data registers</td>
<td>2951</td>
</tr>
<tr>
<td>Table 693</td>
<td>Packet request</td>
<td>2953</td>
</tr>
<tr>
<td>Table 694</td>
<td>ACK response</td>
<td>2953</td>
</tr>
<tr>
<td>Table 695</td>
<td>Data transfer</td>
<td>2953</td>
</tr>
<tr>
<td>Table 696</td>
<td>Debug port registers</td>
<td>2954</td>
</tr>
<tr>
<td>Table 697</td>
<td>Debug port register map and reset values</td>
<td>2960</td>
</tr>
<tr>
<td>Table 698</td>
<td>MEM-AP registers</td>
<td>2962</td>
</tr>
<tr>
<td>Table 699</td>
<td>Access port register map and reset values</td>
<td>2968</td>
</tr>
<tr>
<td>Table 700</td>
<td>System ROM table</td>
<td>2970</td>
</tr>
<tr>
<td>Table 701</td>
<td>MCU ROM table</td>
<td>2970</td>
</tr>
<tr>
<td>Table 702</td>
<td>Processor ROM table</td>
<td>2970</td>
</tr>
<tr>
<td>Table 703</td>
<td>System ROM table register map and reset values</td>
<td>2977</td>
</tr>
<tr>
<td>Table 704</td>
<td>MCU ROM table register map and reset values</td>
<td>2983</td>
</tr>
<tr>
<td>Table 705</td>
<td>CPU ROM table register map and reset values</td>
<td>2988</td>
</tr>
<tr>
<td>Table 706</td>
<td>DWT register map and reset values</td>
<td>3004</td>
</tr>
<tr>
<td>Table 707</td>
<td>ITM register map and reset values</td>
<td>3016</td>
</tr>
<tr>
<td>Table 708</td>
<td>BPU register map and reset values</td>
<td>3024</td>
</tr>
<tr>
<td>Table 709</td>
<td>ETM register map and reset values</td>
<td>3051</td>
</tr>
<tr>
<td>Table 710</td>
<td>TPIU register map and reset values</td>
<td>3066</td>
</tr>
<tr>
<td>Table 711</td>
<td>CTI inputs</td>
<td>3068</td>
</tr>
<tr>
<td>Table 712</td>
<td>CTI outputs</td>
<td>3068</td>
</tr>
<tr>
<td>Table 713</td>
<td>CTI register map and reset values</td>
<td>3080</td>
</tr>
<tr>
<td>Table 714</td>
<td>Peripheral clock freeze control bits</td>
<td>3083</td>
</tr>
<tr>
<td>Table 715</td>
<td>Peripheral behaviour in debug mode</td>
<td>3084</td>
</tr>
<tr>
<td>Table 716</td>
<td>DBGMCU register map and reset values</td>
<td>3099</td>
</tr>
<tr>
<td>Table 717</td>
<td>Document revision history</td>
<td>3106</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Figure 1</td>
<td>System architecture</td>
<td>104</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Memory map based on IDAU mapping</td>
<td>112</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Secure/non-secure partitioning using TrustZone technology</td>
<td>124</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Sharing memory map between CPU in secure and non-secure state</td>
<td>126</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Secure world transition and memory partitioning</td>
<td>126</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Global TrustZone framework and TrustZone awareness</td>
<td>128</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Flash memory TrustZone protections</td>
<td>131</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Flash memory secure HDP area</td>
<td>139</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Key management principle</td>
<td>147</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Device life-cycle security</td>
<td>150</td>
</tr>
<tr>
<td>Figure 11</td>
<td>PRODUCT_STATEs (simplified TrustZone activated view)</td>
<td>153</td>
</tr>
<tr>
<td>Figure 12</td>
<td>PRODUCT_STATEs (full TrustZone activated view)</td>
<td>154</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Collaborative development principle</td>
<td>156</td>
</tr>
<tr>
<td>Figure 14</td>
<td>External Flash memory protection using SFI</td>
<td>158</td>
</tr>
<tr>
<td>Figure 15</td>
<td>GTZC in Armv8-M subsystem block diagram</td>
<td>165</td>
</tr>
<tr>
<td>Figure 16</td>
<td>GTZC block diagram</td>
<td>167</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Watermark memory protection controller (region x/subregions A and B)</td>
<td>169</td>
</tr>
<tr>
<td>Figure 18</td>
<td>MPCBB block diagram</td>
<td>170</td>
</tr>
<tr>
<td>Figure 19</td>
<td>SRAM1, SRAM2 with ECC and SRAM3 with ECC memory map</td>
<td>224</td>
</tr>
<tr>
<td>Figure 20</td>
<td>FLASH block diagram (simplified)</td>
<td>235</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Embedded flash memory organization (2-Mbyte devices)</td>
<td>237</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Embedded flash memory usage</td>
<td>238</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Flash high-cycle data memory map on 2 Mbyte device</td>
<td>251</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Flash high-cycle data memory map on 1 Mbyte device</td>
<td>251</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Flash bank swapping sequence</td>
<td>253</td>
</tr>
<tr>
<td>Figure 26</td>
<td>OBK protection checks</td>
<td>265</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Key writing flow</td>
<td>266</td>
</tr>
<tr>
<td>Figure 28</td>
<td>Swap workflow</td>
<td>268</td>
</tr>
<tr>
<td>Figure 29</td>
<td>HDP in user flash memory</td>
<td>273</td>
</tr>
<tr>
<td>Figure 30</td>
<td>Protection attributes in case of bank swap illustration</td>
<td>278</td>
</tr>
<tr>
<td>Figure 31</td>
<td>ICACHE block diagram</td>
<td>357</td>
</tr>
<tr>
<td>Figure 32</td>
<td>ICACHE TAG and data memories functional view</td>
<td>359</td>
</tr>
<tr>
<td>Figure 33</td>
<td>ICACHE remapping address mechanism</td>
<td>362</td>
</tr>
<tr>
<td>Figure 34</td>
<td>DCACHE block diagram</td>
<td>374</td>
</tr>
<tr>
<td>Figure 35</td>
<td>DCACHE TAG and data memories functional view</td>
<td>377</td>
</tr>
<tr>
<td>Figure 36</td>
<td>Power supply with SMPS</td>
<td>393</td>
</tr>
<tr>
<td>Figure 37</td>
<td>Power supply with LDO</td>
<td>394</td>
</tr>
<tr>
<td>Figure 38</td>
<td>System supply configurations</td>
<td>396</td>
</tr>
<tr>
<td>Figure 39</td>
<td>Power-on (POR) / power-down (PDR) reset waveform</td>
<td>401</td>
</tr>
<tr>
<td>Figure 40</td>
<td>BOR thresholds</td>
<td>402</td>
</tr>
<tr>
<td>Figure 41</td>
<td>PVD thresholds</td>
<td>403</td>
</tr>
<tr>
<td>Figure 42</td>
<td>AVD thresholds</td>
<td>404</td>
</tr>
<tr>
<td>Figure 43</td>
<td>VBAT thresholds</td>
<td>405</td>
</tr>
<tr>
<td>Figure 44</td>
<td>Temperature thresholds</td>
<td>406</td>
</tr>
<tr>
<td>Figure 45</td>
<td>Dynamic voltage scaling in Run mode</td>
<td>407</td>
</tr>
<tr>
<td>Figure 46</td>
<td>I/O states in Standby mode</td>
<td>417</td>
</tr>
<tr>
<td>Figure 47</td>
<td>Simplified diagram of the reset circuit</td>
<td>441</td>
</tr>
<tr>
<td>Figure 48</td>
<td>Clock tree</td>
<td>444</td>
</tr>
</tbody>
</table>
Figure 49. HSE/LSE clock sources ......................................................... 445
Figure 50. CSI calibration flow .......................................................... 448
Figure 51. PLL block diagram ............................................................ 449
Figure 52. PLLs initialization flow ......................................................... 452
Figure 53. CRS block diagram ............................................................ 550
Figure 54. CRS counter behavior .......................................................... 552
Figure 55. Structure of three-volt or five-volt tolerant GPIO (TT or FT) .... 562
Figure 56. Input floating/pull-up/pull-down configurations ....................... 566
Figure 57. Output configuration ............................................................ 567
Figure 58. Alternate function configuration ........................................... 568
Figure 59. High-impedance analog configuration .................................... 569
Figure 60. SBS block diagram .............................................................. 582
Figure 61. Compensation cell management ........................................... 584
Figure 62. Compensation cell usage ..................................................... 585
Figure 63. SBS boot control ................................................................. 586
Figure 64. SBS debug control ............................................................... 589
Figure 65. SBS hardware secure storage control ..................................... 591
Figure 66. GPDMA block diagram ......................................................... 628
Figure 67. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0) .......................................................... 629
Figure 68. GPDMA channel suspend and resume sequence ....................... 630
Figure 69. GPDMA channel abort and restart sequence ........................... 631
Figure 70. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x .................................................. 633
Figure 71. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x .................................................. 634
Figure 72. GPDMA dynamic linked-list data structure of a linear addressing channel x .................................................. 635
Figure 73. GPDMA dynamic linked-list data structure of a 2D addressing channel x .................................................. 635
Figure 74. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0) ....................... 637
Figure 75. Inserting a LLIn with an auxiliary GPDMA channel y .................. 639
Figure 76. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1) ........................... 641
Figure 77. Building LLIn+1: GPDMA dynamic linked-lists in link step mode .................................................. 642
Figure 78. Replace with a new LLIn’ in register file in link step mode ............ 643
Figure 79. Replace with a new LLIn’ and LLIn+1’ in memory in link step mode (option 1) .................................................. 644
Figure 80. Replace with a new LLIn’ and LLIn+1’ in memory in link step mode (option 2) .................................................. 645
Figure 81. GPDMA channel execution and linked-list programming ............... 647
Figure 82. Programmed 2D addressing .................................................... 650
Figure 83. GPDMA arbitration policy ....................................................... 657
Figure 84. Trigger hit, memorization and overrun waveform ......................... 660
Figure 85. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel .................................................. 661
Figure 86. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel .................................................. 662
Figure 87. EXTI block diagram .............................................................. 708
Figure 88. Configurable event trigger logic CPU wakeup .......................... 711
Figure 89. EXTI direct events ................................................................. 712
Figure 90. EXTI mux GPIO selection ...................................................... 713
Figure 91. CRC calculation unit block diagram ....................................... 744
Figure 92. CORDIC convergence for trigonometric functions ....................... 758
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>93</td>
<td>CORDIC convergence for hyperbolic functions</td>
<td>759</td>
</tr>
<tr>
<td>94</td>
<td>CORDIC convergence for square root</td>
<td>760</td>
</tr>
<tr>
<td>95</td>
<td>Block diagram</td>
<td>769</td>
</tr>
<tr>
<td>96</td>
<td>Input buffer areas</td>
<td>771</td>
</tr>
<tr>
<td>97</td>
<td>Circular input buffer</td>
<td>772</td>
</tr>
<tr>
<td>98</td>
<td>Circular input buffer operation</td>
<td>773</td>
</tr>
<tr>
<td>99</td>
<td>Circular output buffer</td>
<td>774</td>
</tr>
<tr>
<td>100</td>
<td>Circular output buffer operation</td>
<td>775</td>
</tr>
<tr>
<td>101</td>
<td>FIR filter structure</td>
<td>777</td>
</tr>
<tr>
<td>102</td>
<td>IIR filter structure (direct form 1)</td>
<td>779</td>
</tr>
<tr>
<td>103</td>
<td>X1 buffer initialization</td>
<td>784</td>
</tr>
<tr>
<td>104</td>
<td>Filtering example 1</td>
<td>785</td>
</tr>
<tr>
<td>105</td>
<td>Filtering example 2</td>
<td>786</td>
</tr>
<tr>
<td>106</td>
<td>FMC block diagram</td>
<td>798</td>
</tr>
<tr>
<td>107</td>
<td>FMC memory banks</td>
<td>800</td>
</tr>
<tr>
<td>108</td>
<td>Mode 1 read access waveforms</td>
<td>810</td>
</tr>
<tr>
<td>109</td>
<td>Mode 1 write access waveforms</td>
<td>810</td>
</tr>
<tr>
<td>110</td>
<td>Mode A read access waveforms</td>
<td>812</td>
</tr>
<tr>
<td>111</td>
<td>Mode A write access waveforms</td>
<td>812</td>
</tr>
<tr>
<td>112</td>
<td>Mode 2 and mode B read access waveforms</td>
<td>814</td>
</tr>
<tr>
<td>113</td>
<td>Mode 2 write access waveforms</td>
<td>815</td>
</tr>
<tr>
<td>114</td>
<td>Mode B write access waveforms</td>
<td>815</td>
</tr>
<tr>
<td>115</td>
<td>Mode C read access waveforms</td>
<td>817</td>
</tr>
<tr>
<td>116</td>
<td>Mode C write access waveforms</td>
<td>818</td>
</tr>
<tr>
<td>117</td>
<td>Mode D read access waveforms</td>
<td>820</td>
</tr>
<tr>
<td>118</td>
<td>Mode D write access waveforms</td>
<td>821</td>
</tr>
<tr>
<td>119</td>
<td>Muxed read access waveforms</td>
<td>823</td>
</tr>
<tr>
<td>120</td>
<td>Muxed write access waveforms</td>
<td>824</td>
</tr>
<tr>
<td>121</td>
<td>Asynchronous wait during a read access waveforms</td>
<td>826</td>
</tr>
<tr>
<td>122</td>
<td>Asynchronous wait during a write access waveforms</td>
<td>827</td>
</tr>
<tr>
<td>123</td>
<td>Wait configuration waveforms</td>
<td>829</td>
</tr>
<tr>
<td>124</td>
<td>Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)</td>
<td>830</td>
</tr>
<tr>
<td>125</td>
<td>Synchronous multiplexed write mode waveforms - PSRAM (CRAM)</td>
<td>832</td>
</tr>
<tr>
<td>126</td>
<td>NAND flash controller waveforms for common memory access</td>
<td>844</td>
</tr>
<tr>
<td>127</td>
<td>Access to non 'CE don't care' NAND-flash</td>
<td>846</td>
</tr>
<tr>
<td>128</td>
<td>Burst write SDRAM access waveforms</td>
<td>855</td>
</tr>
<tr>
<td>129</td>
<td>Burst read SDRAM access waveforms</td>
<td>856</td>
</tr>
<tr>
<td>130</td>
<td>Logic diagram of Read access with RBURST bit set (CAS=1, RPIPE=0)</td>
<td>857</td>
</tr>
<tr>
<td>131</td>
<td>Read access crossing row boundary</td>
<td>859</td>
</tr>
<tr>
<td>132</td>
<td>Write access crossing row boundary</td>
<td>859</td>
</tr>
<tr>
<td>133</td>
<td>Self-refresh mode</td>
<td>861</td>
</tr>
<tr>
<td>134</td>
<td>Power-down mode</td>
<td>862</td>
</tr>
<tr>
<td>135</td>
<td>OCTOSPI block diagram in octal configuration</td>
<td>874</td>
</tr>
<tr>
<td>136</td>
<td>OCTOSPI block diagram in quad configuration</td>
<td>874</td>
</tr>
<tr>
<td>137</td>
<td>OCTOSPI block diagram in dual-quad configuration</td>
<td>875</td>
</tr>
<tr>
<td>138</td>
<td>SDR read command in octal configuration</td>
<td>876</td>
</tr>
<tr>
<td>139</td>
<td>DTR read in octal-SPI mode with DQS (Macronix mode) example</td>
<td>879</td>
</tr>
<tr>
<td>140</td>
<td>SDR write command in octo-SPI mode example</td>
<td>881</td>
</tr>
<tr>
<td>141</td>
<td>DTR write in octal-SPI mode (Macronix mode) example</td>
<td>882</td>
</tr>
<tr>
<td>142</td>
<td>Example of HyperBus read operation</td>
<td>883</td>
</tr>
<tr>
<td>143</td>
<td>HyperBus write operation with initial latency</td>
<td>885</td>
</tr>
<tr>
<td>144</td>
<td>HyperBus read operation with additional latency</td>
<td>885</td>
</tr>
</tbody>
</table>
List of figures RM0481

Figure 197. Stopping ongoing regular conversions ................................................. 1036
Figure 198. Stopping ongoing regular and injected conversions ................................. 1037
Figure 199. Triggers shared between ADC master and slave ...................................... 1038
Figure 200. Injected conversion latency ................................................................. 1040
Figure 201. Example of JSQR queue of context (sequence change) ............................. 1043
Figure 202. Example of JSQR queue of context (trigger change) ............................... 1043
Figure 203. Example of JSQR queue of context with overflow before conversion ........... 1044
Figure 204. Example of JSQR queue of context with overflow during conversion .......... 1044
Figure 205. Example of JSQR queue of context with empty queue (case JQM = 0) ...... 1045
Figure 206. Example of JSQR queue of context with empty queue (JQM = 1) ............ 1046
Figure 207. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion ................................................. 1046
Figure 208. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs ....................... 1047
Figure 209. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion ............................................... 1047
Figure 210. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 1) ............ 1048
Figure 211. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 0) ............. 1048
Figure 212. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 1) ............. 1049
Figure 213. Single conversions of a sequence, software trigger .................................. 1051
Figure 214. Continuous conversion of a sequence, software trigger ............................ 1051
Figure 215. Single conversions of a sequence, hardware trigger ................................ 1052
Figure 216. Continuous conversions of a sequence, hardware trigger .......................... 1052
Figure 217. Right alignment (offset disabled, unsigned value) ................................... 1054
Figure 218. Right alignment (offset enabled, signed value) ....................................... 1055
Figure 219. Left alignment (offset disabled, unsigned value) ..................................... 1055
Figure 220. Left alignment (offset enabled, signed value) ........................................ 1056
Figure 221. Example of overrun (OVRMOD = 0) ...................................................... 1057
Figure 222. Example of overrun (OVRMOD = 1) ...................................................... 1058
Figure 223. AUTODLY = 1, regular conversion in continuous mode, software trigger ........ 1061
Figure 224. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0) ................................................................. 1061
Figure 225. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1; JDISCEN = 1) ................................................................. 1062
Figure 226. AUTODLY = 1, regular continuous conversions interrupted by injected conversions .... 1063
Figure 227. AUTODLY = 1 in auto-injected mode (JAUTO = 1) ............................. 1063
Figure 228. Analog watchdog guarded area ............................................................. 1064
Figure 229. ADCy_AWDx_OUT signal generation (on all regular channels) ............... 1066
Figure 230. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software) .... 1067
Figure 231. ADCy_AWDx_OUT signal generation (on a single regular channel) .......... 1067
Figure 232. ADCy_AWDx_OUT signal generation (on all injected channels) ............... 1067
Figure 233. 20-bit to 16-bit result truncation .......................................................... 1069
Figure 234. Numerical example with 5-bit shift and rounding .................................... 1069
Figure 235. Triggered regular oversampling mode (TROVS bit = 1) .......................... 1071
Figure 236. Regular oversampling modes (4x ratio) ................................................. 1072
Figure 237. Regular and injected oversampling modes used simultaneously ................. 1073
Figure 238. Triggered regular oversampling with injection ....................................... 1073
Figure 239. Oversampling in auto-injected mode ..................................................... 1074
Figure 240. Dual ADC block diagram(1) ................................................................. 1076
Figure 241. Injected simultaneous mode on 4 channels: dual ADC mode ....................... 1077
Figure 242. Regular simultaneous mode on 16 channels: dual ADC mode .................... 1079
Figure 292. Encryption/decryption typical usage.
Figure 290. RNG initialization overview.
Figure 289. NIST SP800-90B entropy source model.
Figure 288. RNG initialization overview.
Figure 287. AES block diagram.
Figure 286. Bidirectional PSSI_DE/PSSI_RDY waveform.
Figure 285. Data enable waveform diagram in receive mode (CKPOL=0).
Figure 284. Data enable waveform diagram in transmit mode (CKPOL=0).
Figure 283. Data enable in receive mode waveform diagram (CKPOL=0).
Figure 282. DCMI signal waveforms.
Figure 281. PSSI block diagram.
Figure 280. Pixel raster scan order.
Figure 279. Data capture waveforms.
Figure 278. Coordinates and size of the window after cropping.
Figure 277. Frame capture waveforms in continuous grab mode.
Figure 276. Frame capture waveforms in snapshot mode.
Figure 275. Timing diagram.
Figure 274. DCMI block diagram.
Figure 273. VREFBUF block diagram.
Figure 272. DAC Sample and hold mode phase diagram.
Figure 271. DAC conversion (SW trigger enabled) with triangle wave generation.
Figure 270. DAC triangle wave generation.
Figure 269. DAC conversion (SW trigger enabled). - case 2: Slave interrupted first.
Figure 268. DAC conversion (SW trigger enabled) with LFSR wave generation.
Figure 267. ADC LFSR register calculation algorithm.
Figure 266. Timing diagram for conversion with trigger disabled TEN = 0.
Figure 265. Data registers in dual DAC channel mode.
Figure 264. Data registers in single DAC channel mode.
Figure 263. Dual-channel DAC block diagram.
Figure 262. Temperature sensor functional block diagram.
Figure 261. Method for high REF_CLK frequencies.
Figure 260. Method for low REF_CLK frequencies.
Figure 259. Temperature sensor block diagram.
Figure 258. VREFINT channel block diagram.
Figure 257. VBAT channel block diagram.
Figure 256. Temperature sensor channel block diagram.
Figure 255. DMA requests in interleaved mode when MDMA = 0b10.
Figure 254. DMA requests in regular simultaneous mode when MDMA = 0b10.
Figure 253. DMA Requests in regular simultaneous mode when MDMA = 0b00.
Figure 252. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12.
Figure 251. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12.
Figure 250. Interleaved single channel CH0 with injected sequence CH11, CH12.
Figure 249. Case of trigger occurring during injected conversion.
Figure 248. Alternate + regular simultaneous.
Figure 247. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode.
Figure 246. Alternate trigger: injected group of each ADC.
Figure 245. Interleaved conversion with injection.
Figure 244. Interleaved mode on 1 channel in single conversion mode: dual ADC mode.
Figure 243. Interleaved mode on one channel in continuous conversion mode: dual ADC mode.
RM0481 Rev 1 89/3119
Figure 344. Counter timing diagram, update event when ARPE=0

Figure 343. Counter timing diagram, internal clock divided by N

Figure 342. Counter timing diagram, internal clock divided by 4

Figure 341. Counter timing diagram, internal clock divided by 2

Figure 340. Counter timing diagram, internal clock divided by 1

Figure 339. Counter timing diagram, update event when ARPE=0
Figure 345. Counter timing diagram, update event when ARPE=1
(TIMx_ARR not preloaded) ................................................................. 1438
Figure 346. Counter timing diagram, internal clock divided by 1  . 1440
Figure 347. Counter timing diagram, internal clock divided by 2 . 1441
Figure 348. Counter timing diagram, internal clock divided by 4 . 1441
Figure 349. Counter timing diagram, internal clock divided by N . 1442
Figure 350. Counter timing diagram, update event when repetition counter is not used . 1442
Figure 351. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . 1444
Figure 352. Counter timing diagram, internal clock divided by 2 . 1444
Figure 353. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . 1445
Figure 354. Counter timing diagram, internal clock divided by N . 1445
Figure 355. Counter timing diagram, update event with ARPE=1 (counter underflow) . 1446
Figure 356. Counter timing diagram, Update event with ARPE=1 (counter overflow) . 1447
Figure 357. Update rate examples depending on mode and TIMx_RCR register settings . 1448
Figure 358. External trigger input block . 1449
Figure 359. Control circuit in normal mode, internal clock divided by 1 . 1450
Figure 360. tim_t12 external clock connection example . 1450
Figure 361. Control circuit in external clock mode 1 . 1451
Figure 362. External trigger input block . 1452
Figure 363. Control circuit in external clock mode 2 . 1453
Figure 364. Capture/compare channel (example: channel 1 input stage) . 1453
Figure 365. Capture/compare channel 1 main circuit . 1454
Figure 366. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4) . 1455
Figure 367. Output stage of capture/compare channel (channel 5, idem ch. 6) . 1455
Figure 368. PWM input mode timing . 1458
Figure 369. Output compare mode, toggle on tim_oc1 . 1460
Figure 370. Edge-aligned PWM waveforms (ARR=8) . 1461
Figure 371. Center-aligned PWM waveforms (ARR=8) . 1462
Figure 372. Dithering principle . 1463
Figure 373. Data format and register coding in dithering mode . 1464
Figure 374. PWM resolution vs frequency . 1465
Figure 375. PWM dithering pattern . 1466
Figure 376. Dithering effect on duty cycle in center-aligned PWM mode . 1467
Figure 377. Generation of 2 phase-shifted PWM signals with 50% duty cycle . 1469
Figure 378. Combined PWM mode on channel 1 and 3 . 1470
Figure 379. 3-phase combined PWM signals with multiple trigger pulses per period . 1471
Figure 380. Complementary output with symmetrical dead-time insertion . 1472
Figure 381. Asymmetrical deadtime . 1473
Figure 382. Dead-time waveforms with delay greater than the negative pulse . 1473
Figure 383. Dead-time waveforms with delay greater than the positive pulse . 1473
Figure 384. Break and Break2 circuitry overview . 1476
Figure 385. Various output behavior in response to a break event on tim_brk (OSSI = 1) . 1478
Figure 386. PWM output state following tim_brk and tim_brk2 assertion (OSSI=1) . 1479
Figure 387. PWM output state following tim_brk assertion (OSSI=0) . 1480
Figure 388. Output redirection (tim_brk2 request not represented) . 1481
Figure 389. tim_ocref_clr input selection multiplexer . 1482
Figure 390. Clearing TIMx_tim_ocxref . 1483
Figure 391. 6-step generation, COM example (OSSR=1) . 1484
Figure 392. Example of one pulse mode . 1485
Figure 393. Retriggerable one-pulse mode . 1486
Figure 394. Pulse generator circuitry . 1487
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>395</td>
<td>Pulse generation on compare event, for edge-aligned and encoder modes</td>
<td>1488</td>
</tr>
<tr>
<td>396</td>
<td>Extended pulselwidth in case of concurrent triggers</td>
<td>1489</td>
</tr>
<tr>
<td>397</td>
<td>Example of counter operation in encoder interface mode.</td>
<td>1491</td>
</tr>
<tr>
<td>398</td>
<td>Example of encoder interface mode with TIMx_T1fp1 polarity inverted.</td>
<td>1491</td>
</tr>
<tr>
<td>399</td>
<td>Quadrature encoder counting modes</td>
<td>1492</td>
</tr>
<tr>
<td>400</td>
<td>Direction plus clock encoder mode</td>
<td>1493</td>
</tr>
<tr>
<td>401</td>
<td>Directional clock encoder mode (CC1P = CC2P = 0)</td>
<td>1493</td>
</tr>
<tr>
<td>402</td>
<td>Directional clock encoder mode (CC1P = CC2P = 1)</td>
<td>1494</td>
</tr>
<tr>
<td>403</td>
<td>Index gating options</td>
<td>1495</td>
</tr>
<tr>
<td>404</td>
<td>Jittered Index signals</td>
<td>1496</td>
</tr>
<tr>
<td>405</td>
<td>Index generation for IPOS[1:0] = 11</td>
<td>1497</td>
</tr>
<tr>
<td>406</td>
<td>Counter reading with index gated on channel A (IPOS[1:0] = 11)</td>
<td>1497</td>
</tr>
<tr>
<td>407</td>
<td>Counter reading with index ungated (IPOS[1:0] = 00)</td>
<td>1497</td>
</tr>
<tr>
<td>408</td>
<td>Counter reading with index gated on channel A and B.</td>
<td>1498</td>
</tr>
<tr>
<td>409</td>
<td>Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)</td>
<td>1499</td>
</tr>
<tr>
<td>410</td>
<td>Encoder reset Narrow index pulse (closer view, ARR = 0x07)</td>
<td>1500</td>
</tr>
<tr>
<td>411</td>
<td>Index behavior in x1 and x2 mode (IPOS[1:0] = 01).</td>
<td>1501</td>
</tr>
<tr>
<td>412</td>
<td>Directional index sensitivity</td>
<td>1501</td>
</tr>
<tr>
<td>413</td>
<td>Counter reset as function of FIDX bit setting</td>
<td>1502</td>
</tr>
<tr>
<td>414</td>
<td>Index blanking.</td>
<td>1502</td>
</tr>
<tr>
<td>415</td>
<td>Index behavior in clock + direction mode, IPOS[0] = 1</td>
<td>1503</td>
</tr>
<tr>
<td>416</td>
<td>Index behavior in directional clock mode, IPOS[0] = 1</td>
<td>1503</td>
</tr>
<tr>
<td>417</td>
<td>State diagram for quadrature encoded signals</td>
<td>1504</td>
</tr>
<tr>
<td>418</td>
<td>Up-counting encoder error detection</td>
<td>1505</td>
</tr>
<tr>
<td>419</td>
<td>Down-counting encoder error detection</td>
<td>1506</td>
</tr>
<tr>
<td>420</td>
<td>Encoder mode change with preload transferred on update (SMSPS = 0)</td>
<td>1507</td>
</tr>
<tr>
<td>421</td>
<td>Measuring time interval between edges on 3 signals</td>
<td>1508</td>
</tr>
<tr>
<td>422</td>
<td>Example of Hall sensor interface</td>
<td>1510</td>
</tr>
<tr>
<td>423</td>
<td>Control circuit in reset mode</td>
<td>1511</td>
</tr>
<tr>
<td>424</td>
<td>Control circuit in Gated mode</td>
<td>1512</td>
</tr>
<tr>
<td>425</td>
<td>Control circuit in trigger mode</td>
<td>1513</td>
</tr>
<tr>
<td>426</td>
<td>Control circuit in external clock mode 2 + trigger mode</td>
<td>1514</td>
</tr>
<tr>
<td>427</td>
<td>General-purpose timer block diagram</td>
<td>1572</td>
</tr>
<tr>
<td>428</td>
<td>Counter timing diagram with prescaler division change from 1 to 2</td>
<td>1577</td>
</tr>
<tr>
<td>429</td>
<td>Counter timing diagram with prescaler division change from 1 to 4</td>
<td>1578</td>
</tr>
<tr>
<td>430</td>
<td>Counter timing diagram, internal clock divided by 1</td>
<td>1579</td>
</tr>
<tr>
<td>431</td>
<td>Counter timing diagram, internal clock divided by 2</td>
<td>1579</td>
</tr>
<tr>
<td>432</td>
<td>Counter timing diagram, internal clock divided by 4</td>
<td>1580</td>
</tr>
<tr>
<td>433</td>
<td>Counter timing diagram, internal clock divided by N</td>
<td>1580</td>
</tr>
<tr>
<td>434</td>
<td>Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)</td>
<td>1581</td>
</tr>
<tr>
<td>435</td>
<td>Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)</td>
<td>1582</td>
</tr>
<tr>
<td>436</td>
<td>Counter timing diagram, internal clock divided by 1</td>
<td>1583</td>
</tr>
<tr>
<td>437</td>
<td>Counter timing diagram, internal clock divided by 2</td>
<td>1584</td>
</tr>
<tr>
<td>438</td>
<td>Counter timing diagram, internal clock divided by 4</td>
<td>1584</td>
</tr>
<tr>
<td>439</td>
<td>Counter timing diagram, internal clock divided by N</td>
<td>1585</td>
</tr>
<tr>
<td>440</td>
<td>Counter timing diagram, Update event</td>
<td>1585</td>
</tr>
<tr>
<td>441</td>
<td>Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x8</td>
<td>1587</td>
</tr>
<tr>
<td>442</td>
<td>Counter timing diagram, internal clock divided by 2</td>
<td>1587</td>
</tr>
<tr>
<td>443</td>
<td>Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36</td>
<td>1588</td>
</tr>
<tr>
<td>444</td>
<td>Counter timing diagram, internal clock divided by N</td>
<td>1588</td>
</tr>
<tr>
<td>445</td>
<td>Counter timing diagram, Update event with ARPE=1 (counter underflow)</td>
<td>1589</td>
</tr>
<tr>
<td>446</td>
<td>Counter timing diagram, Update event with ARPE=1 (counter overflow)</td>
<td>1590</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>509.</td>
<td>Basic timer block diagram</td>
<td></td>
</tr>
<tr>
<td>510.</td>
<td>Control circuit in normal mode, internal clock divided by 1</td>
<td></td>
</tr>
<tr>
<td>511.</td>
<td>Counter timing diagram with prescaler division change from 1 to 2</td>
<td></td>
</tr>
<tr>
<td>512.</td>
<td>Counter timing diagram with prescaler division change from 1 to 4</td>
<td></td>
</tr>
<tr>
<td>513.</td>
<td>Counter timing diagram, internal clock divided by 1</td>
<td></td>
</tr>
<tr>
<td>514.</td>
<td>Counter timing diagram, internal clock divided by 2</td>
<td></td>
</tr>
<tr>
<td>515.</td>
<td>Counter timing diagram, internal clock divided by 4</td>
<td></td>
</tr>
<tr>
<td>516.</td>
<td>Counter timing diagram, internal clock divided by N</td>
<td></td>
</tr>
<tr>
<td>517.</td>
<td>Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)</td>
<td></td>
</tr>
<tr>
<td>518.</td>
<td>Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)</td>
<td></td>
</tr>
<tr>
<td>519.</td>
<td>Dithering principle</td>
<td></td>
</tr>
<tr>
<td>520.</td>
<td>Data format and register coding in dithering mode</td>
<td></td>
</tr>
<tr>
<td>521.</td>
<td>FCnt resolution vs frequency</td>
<td></td>
</tr>
<tr>
<td>522.</td>
<td>PWM dithering pattern</td>
<td></td>
</tr>
<tr>
<td>523.</td>
<td>General-purpose timer block diagram (TIM12)</td>
<td></td>
</tr>
<tr>
<td>524.</td>
<td>General-purpose timer block diagram (TIM13/TIM14)</td>
<td></td>
</tr>
<tr>
<td>525.</td>
<td>Counter timing diagram with prescaler division change from 1 to 2</td>
<td></td>
</tr>
<tr>
<td>526.</td>
<td>Counter timing diagram with prescaler division change from 1 to 4</td>
<td></td>
</tr>
<tr>
<td>527.</td>
<td>Counter timing diagram, internal clock divided by 1</td>
<td></td>
</tr>
<tr>
<td>528.</td>
<td>Counter timing diagram, internal clock divided by 2</td>
<td></td>
</tr>
<tr>
<td>529.</td>
<td>Counter timing diagram, internal clock divided by 4</td>
<td></td>
</tr>
<tr>
<td>530.</td>
<td>Counter timing diagram, internal clock divided by N</td>
<td></td>
</tr>
<tr>
<td>531.</td>
<td>Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)</td>
<td></td>
</tr>
<tr>
<td>532.</td>
<td>Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)</td>
<td></td>
</tr>
<tr>
<td>533.</td>
<td>Control circuit in normal mode, internal clock divided by 1</td>
<td></td>
</tr>
<tr>
<td>534.</td>
<td>tim_t12 external clock connection example</td>
<td></td>
</tr>
<tr>
<td>535.</td>
<td>Control circuit in external clock mode 1</td>
<td></td>
</tr>
<tr>
<td>536.</td>
<td>Capture/compare channel 1 input stage (TIM13/TIM14)</td>
<td></td>
</tr>
<tr>
<td>537.</td>
<td>Capture/compare channel 1 input stage (TIM12)</td>
<td></td>
</tr>
<tr>
<td>538.</td>
<td>Capture/compare channel 1 main circuit</td>
<td></td>
</tr>
<tr>
<td>539.</td>
<td>Output stage of capture/compare channel 1</td>
<td></td>
</tr>
<tr>
<td>540.</td>
<td>PWM input mode timing</td>
<td></td>
</tr>
<tr>
<td>541.</td>
<td>Output compare mode, toggle on tim_oc1</td>
<td></td>
</tr>
<tr>
<td>542.</td>
<td>Edge-aligned PWM waveforms (ARR=8)</td>
<td></td>
</tr>
<tr>
<td>543.</td>
<td>Dithering principle</td>
<td></td>
</tr>
<tr>
<td>544.</td>
<td>Data format and register coding in dithering mode</td>
<td></td>
</tr>
<tr>
<td>545.</td>
<td>PWM resolution vs frequency</td>
<td></td>
</tr>
<tr>
<td>546.</td>
<td>PWM dithering pattern</td>
<td></td>
</tr>
</tbody>
</table>
Figure 547. Combined PWM mode on channel 1 and 2 ................................................................. 1731
Figure 548. Example of one pulse mode ........................................................................................ 1732
Figure 549. Retriggerable one pulse mode ...................................................................................... 1733
Figure 550. Measuring time interval between edges on 2 signals .................................................. 1734
Figure 551. Control circuit in reset mode ....................................................................................... 1735
Figure 552. Control circuit in gated mode ....................................................................................... 1736
Figure 553. Control circuit in trigger mode ..................................................................................... 1736
Figure 554. TIM15 block diagram ................................................................................................. 1772
Figure 555. TIM16/TIM17 block diagram ...................................................................................... 1773
Figure 556. Counter timing diagram with prescaler division change from 1 to 2 ......................... 1777
Figure 557. Counter timing diagram with prescaler division change from 1 to 4 ......................... 1778
Figure 558. Counter timing diagram, internal clock divided by 1 .................................................. 1779
Figure 559. Counter timing diagram, internal clock divided by 2 .................................................. 1780
Figure 560. Counter timing diagram, internal clock divided by 4 .................................................. 1780
Figure 561. Counter timing diagram, internal clock divided by N .................................................. 1781
Figure 562. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) ................................................................. 1781
Figure 563. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) ........ 1782
Figure 564. Update rate examples depending on mode and TIMx_RCR register settings ............ 1783
Figure 565. Control circuit in normal mode, internal clock divided by 1 ........................................ 1784
Figure 566. tim_t2 external clock connection example ................................................................. 1784
Figure 567. Control circuit in external clock mode 1 ................................................................... 1785
Figure 568. Capture/compare channel (example: channel 1 input stage) ..................................... 1786
Figure 569. Capture/compare channel 1 main circuit ................................................................. 1786
Figure 570. Output stage of capture/compare channel (channel 1) ................................................. 1787
Figure 571. Output stage of capture/compare channel (channel 2 for TIM15) ............................. 1787
Figure 572. PWM input mode timing .............................................................................................. 1790
Figure 573. Output compare mode, toggle on tim_oc1 ................................................................. 1792
Figure 574. Edge-aligned PWM waveforms (ARR=8) ................................................................. 1793
Figure 575. Dithering principle ...................................................................................................... 1794
Figure 576. Data format and register coding in dithering mode .................................................. 1794
Figure 577. PWM resolution vs frequency .................................................................................... 1795
Figure 578. PWM dithering pattern ............................................................................................... 1796
Figure 579. Combined PWM mode on channel 1 and 2 ............................................................... 1798
Figure 580. Complementary output with symmetrical dead-time insertion .................................. 1799
Figure 581. Asymmetrical deadtime .............................................................................................. 1800
Figure 582. Dead-time waveforms with delay greater than the negative pulse ......................... 1800
Figure 583. Dead-time waveforms with delay greater than the positive pulse ......................... 1800
Figure 584. Break circuitry overview ............................................................................................ 1802
Figure 585. Output behavior in response to a break event on tim_brk ........................................... 1804
Figure 586. Output redirection ...................................................................................................... 1806
Figure 587. tim_ocref_clr input selection multiplexer ................................................................. 1807
Figure 588. 6-step generation, COM example (OSSR=1) ............................................................... 1808
Figure 589. Example of one pulse mode ....................................................................................... 1809
Figure 590. Retriggerable one pulse mode .................................................................................... 1811
Figure 591. Measuring time interval between edges on 2 signals ................................................. 1811
Figure 592. Control circuit in reset mode ....................................................................................... 1812
Figure 593. Control circuit in gated mode ..................................................................................... 1813
Figure 594. Control circuit in trigger mode .................................................................................... 1814
Figure 595. LPTIM1/2/3/5/6 timer block diagram(1) .................................................................. 1877
Figure 596. LPTIM4 timer block diagram(1) .............................................................................. 1878
Figure 597. Glitch filter timing diagram ...................................................... 1882
Figure 598. LPTIM output waveform, single counting mode configuration when repetition register content is different than zero (with PRELOAD = 1) ........ 1884
Figure 599. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set) .................................. 1884
Figure 600. LPTIM output waveform, Continuous counting mode configuration .......................................................... 1885
Figure 601. Waveform generation .............................................................. 1886
Figure 602. Encoder mode counting sequence ......................................... 1890
Figure 603. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1) .................................................. 1891
Figure 604. Capture/compare input stage (channel 1) .............................. 1892
Figure 605. Capture/compare output stage (channel 1) ............................ 1892
Figure 606. Edge-aligned PWM mode (PRELOAD = 1) ............................ 1894
Figure 607. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0) ........ 1895
Figure 608. PWM mode with immediate update versus preloaded update .......................................................... 1896
Figure 609. Independent watchdog block diagram .................................. 1926
Figure 610. Reset timing due to timeout .................................................... 1928
Figure 611. Reset timing due to refresh in the not allowed area ................. 1929
Figure 612. Example of window comparator update .................................. 1930
Figure 613. Independent watchdog interrupt timing diagram ................... 1932
Figure 614. Example of early wake-up comparator update ..................... 1933
Figure 615. Watchdog block diagram ...................................................... 1941
Figure 616. Window watchdog timing diagram ........................................ 1943
Figure 617. RTC block diagram .............................................................. 1949
Figure 618. TAMP block diagram ......................................................... 2007
Figure 619. Backup registers protection zones ........................................ 2012
Figure 620. Active tamper filtering ......................................................... 2018
Figure 621. I2C block diagram ............................................................... 2054
Figure 622. I2C bus protocol ................................................................. 2056
Figure 623. Setup and hold timings ........................................................ 2058
Figure 624. I2C initialization flow ........................................................... 2061
Figure 625. Data reception ................................................................. 2062
Figure 626. Data transmission ............................................................. 2063
Figure 627. Slave initialization flow ....................................................... 2066
Figure 628. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 0 .......................................................... 2068
Figure 629. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1 .......................................................... 2069
Figure 630. Transfer bus diagrams for I2C slave transmitter (mandatory events only) .......................................................... 2070
Figure 631. Transfer sequence flow for slave receiver with NOSTRETCH = 0 .......................................................... 2071
Figure 632. Transfer sequence flow for slave receiver with NOSTRETCH = 1 .......................................................... 2072
Figure 633. Transfer bus diagrams for I2C slave receiver (mandatory events only) .......................................................... 2072
Figure 634. Master clock generation ....................................................... 2074
Figure 635. Master initialization flow ..................................................... 2076
Figure 636. 10-bit address read access with HEAD10R = 0 .................... 2076
Figure 637. 10-bit address read access with HEAD10R = 1 .................... 2077
Figure 638. Transfer sequence flow for I2C master transmitter for N ≤ 255 bytes .......................................................... 2078
Figure 639. Transfer sequence flow for I2C master transmitter for N > 255 bytes .......................................................... 2079
Figure 640. Transfer bus diagrams for I2C master transmitter (mandatory events only) .......................................................... 2080
Figure 641. Transfer sequence flow for I2C master receiver for N ≤ 255 bytes .......................................................... 2082
Figure 642. Transfer sequence flow for I2C master receiver for N > 255 bytes .......................................................... 2083
Figure 643. Transfer bus diagrams for I2C master receiver
Figure 644. Timeout intervals for LOW-SEXT, LOW-MEXT .......................................................... 2088
Figure 645. Transfer sequence flow for SMBus slave transmitter N bytes + PEC ...................... 2091
Figure 646. Transfer bus diagrams for SMBus slave transmitter (SBC = 1) ............................... 2092
Figure 647. Transfer sequence flow for SMBus slave receiver N bytes + PEC ......................... 2093
Figure 648. Bus transfer diagrams for SMBus slave receiver (SBC = 1) ................................. 2094
Figure 649. Bus transfer diagrams for SMBus master transmitter ............................................. 2095
Figure 650. Bus transfer diagrams for SMBus master receiver .................................................... 2097
Figure 651. I3C block diagram .................................................................................................. 2123
Figure 652. I3C (primary) controller state and programming sequence diagram ....................... 2127
Figure 653. I3C target state and programming sequence diagram ............................................. 2132
Figure 654. I3C CCC messages, as controller ............................................................................. 2146
Figure 655. I3C broadcast ENTDAA CCC, as controller .......................................................... 2147
Figure 656. I3C broadcast, direct read and direct write RSTACT CCC, as controller ................ 2148
Figure 657. I3C CCC messages, as target .................................................................................... 2150
Figure 658. I3C broadcast ENTDAA CCC, as target ................................................................. 2151
Figure 659. I3C broadcast DEFTGTS CCC, as target ................................................................. 2152
Figure 660. I3C broadcast DEFGPAA CCC, as target ............................................................... 2153
Figure 661. I3C private read/write messages, as controller ........................................................ 2155
Figure 662. I3C private read/write messages, as target ............................................................... 2156
Figure 663. Legacy I2C read/write messages, as controller ......................................................... 2157
Figure 664. IBI transfer, as controller/target ............................................................................... 2158
Figure 665. Hot-join request transfer, as controller/target ........................................................... 2159
Figure 666. Controller-role request transfer, as controller/target ............................................... 2160
Figure 667. C-FIFO management, as controller .......................................................................... 2161
Figure 668. TX-FIFO management, as controller ...................................................................... 2163
Figure 669. RX-FIFO management, as controller ...................................................................... 2165
Figure 670. S-FIFO management, as controller ......................................................................... 2168
Figure 671. RX-FIFO management, as target ............................................................................ 2169
Figure 672. TX-FIFO management with I3C_TGTDDR, as target .............................................. 2171
Figure 673. TX-FIFO management by software without I3C_TGTDDR if reading less bytes than TX-FIFO size, as target ................................................................. 2173
Figure 674. USART block diagram ............................................................................................ 2231
Figure 675. Word length programming ...................................................................................... 2235
Figure 676. Configurable stop bits ............................................................................................. 2237
Figure 677. TC/TXE behavior when transmitting .................................................................... 2239
Figure 678. Start bit detection when oversampling by 16 or 8 .................................................... 2240
Figure 679. user_ker_ck clock divider block diagram .............................................................. 2243
Figure 680. Data sampling when oversampling by 16 ............................................................... 2244
Figure 681. Data sampling when oversampling by 8 ................................................................. 2245
Figure 682. Mute mode using Idle line detection ....................................................................... 2252
Figure 683. Mute mode using address mark detection .............................................................. 2253
Figure 684. Break detection in LIN mode (11-bit break length - LBDL bit is set) ...................... 2256
Figure 685. Break detection in LIN mode vs. Framing error detection ....................................... 2257
Figure 686. USART example of synchronous master transmission ......................................... 2258
Figure 687. USART data clock timing diagram in Synchronous master mode (M bits = 00) .................................................................................................................. 2258
Figure 688. USART data clock timing diagram in Synchronous master mode (M bits = 01) .................................................................................................................. 2259
Figure 689. USART data clock timing diagram in Synchronous slave mode (M bits = 00) .................................................................................................................. 2260
Figure 690. ISO 7816-3 asynchronous protocol ....................................................................... 2262
List of figures

Figure 691. Parity error detection using the 1.5 stop bits ........................................... 2264
Figure 692. IrDA SIR ENDEC block diagram ................................................................. 2268
Figure 693. IrDA data modulation (3/16) - Normal mode ............................................. 2268
Figure 694. Transmission using DMA .......................................................... .......................... 2270
Figure 695. Reception using DMA .................................................... .......................... 2271
Figure 696. Hardware flow control between 2 USARTs .................................................. 2271
Figure 697. RS232 RTS flow control ............................................................................. 2272
Figure 698. RS232 CTS flow control ............................................................................. 2273
Figure 699. Wake-up event verified (wake-up event = address match, FIFO disabled) ........ 2276
Figure 700. Wake-up event not verified (wake-up event = address match, FIFO disabled) .......................................................... 2276
Figure 701. LPUART block diagram ............................................................................. 2319
Figure 702. LPUART word length programming .......................................................... 2322
Figure 703. Configurable stop bits ............................................................................. 2324
Figure 704. TC/TXE behavior when transmitting .......................................................... 2326
Figure 705. Lpurt_ker_ck clock divider block diagram .................................................. 2330
Figure 706. Mute mode using Idle line detection ............................................................ 2335
Figure 707. Mute mode using address mark detection ..................................................... 2336
Figure 708. Transmission using DMA ........................................................................ 2338
Figure 709. Reception using DMA ............................................................................ 2339
Figure 710. Hardware flow control between 2 LPUARTs .................................................. 2340
Figure 711. RS232 RTS flow control ............................................................................. 2340
Figure 712. RS232 CTS flow control ............................................................................. 2341
Figure 713. Wake-up event verified (wake-up event = address match, FIFO disabled) ........ 2344
Figure 714. Wake-up event not verified (wake-up event = address match, FIFO disabled) .......................................................... 2344
Figure 715. SPI/I2S block diagram ............................................................................. 2374
Figure 716. Full-duplex single master/ single slave application ..................................... 2377
Figure 717. Half-duplex single master/ single slave application ..................................... 2378
Figure 718. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode) .......................................................... 2379
Figure 719. Master and three independent slaves connected in star topology ................. 2380
Figure 720. Master and three slaves connected in circular (daisy chain) topology .......... 2381
Figure 721. Multimaster application ............................................................................ 2382
Figure 722. Scheme of SS control logic ..................................................................... 2384
Figure 723. Data flow timing control (SSEO = 1, SSOM = 0, SSM = 0) ....................... 2384
Figure 724. SS interleaving pulses between data (SSEO = 1, SSOM = 1, SSM = 0) .... 2385
Figure 725. Data clock timing diagram ..................................................................... 2387
Figure 726. Data alignment when data size is not equal to 8-, 16- or 32-bit .......... 2388
Figure 727. Packing data in FIFO for transmission and reception at full feature set instance .......................................................... 2396
Figure 728. TI mode transfer ..................................................................................... 2398
Figure 729. Optional configurations of slave detecting underrun condition ................. 2400
Figure 730. Waveform examples ............................................................................. 2407
Figure 731. Master I2S Philips protocol waveforms (16/32-bit full accuracy) .............. 2408
Figure 732. I2S Philips standard waveforms ................................................................. 2408
Figure 733. Master MSB Justified 16-bit or 32-bit full-accuracy length ................. 2409
Figure 734. Master MSB justified 16 or 24-bit data length .......................................... 2409
Figure 735. Slave MSB justified ............................................................................. 2410
Figure 736. LSB justified 16 or 24-bit data length ......................................................... 2410
Figure 737. Master PCM when the frame length is equal the data length ................. 2411
Figure 738. Master PCM standard waveforms (16 or 24-bit data length) ................. 2411

98/3119 RM0481 Rev 1
Figure 739. Slave PCM waveforms .................................................. 2412
Figure 740. Startup sequence, I2S Philips standard, master. ................. 2415
Figure 741. Startup sequence, I2S Philips standard, slave ................. 2416
Figure 742. Stop sequence, I2S Philips standard, master ................. 2416
Figure 743. I^2S clock generator architecture ............................... 2417
Figure 744. Data Format .............................................................. 2419
Figure 745. Handling of underrun situation ................................. 2421
Figure 746. Handling of overrun situation ................................... 2422
Figure 747. Frame error detection, with FIXCH=0 ......................... 2423
Figure 748. Frame error detection, with FIXCH=1 ......................... 2423
Figure 749. SAI functional block diagram ................................... 2450
Figure 750. Audio frame .............................................................. 2454
Figure 751. FS role is start of frame + channel side identification (FSDEF = TRIS = 1) .................................................. 2456
Figure 752. FS role is start of frame (FSDEF = 0) .......................... 2457
Figure 753. Slot size configuration with FBOFF = 0 in SAI_xSLOT R .... 2458
Figure 754. First bit offset ............................................................. 2458
Figure 755. Audio block clock generator overview ......................... 2460
Figure 756. PDM typical connection and timing ............................ 2464
Figure 757. Detailed PDM interface block diagram ......................... 2465
Figure 758. Start-up sequence ..................................................... 2466
Figure 759. SAI_ADR format in TDM, 32-bit slot width .................. 2467
Figure 760. SAI_ADR format in TDM, 16-bit slot width .................. 2468
Figure 761. SAI_ADR format in TDM, 8-bit slot width .................. 2469
Figure 762. AC’97 audio frame ...................................................... 2472
Figure 763. Example of typical AC’97 configuration on devices featuring at least 2 embedded SAlS (three external AC’97 decoders) .......................... 2473
Figure 764. SPDIF format ............................................................. 2474
Figure 765. SAI_xDR register ordering ......................................... 2475
Figure 766. Data companding hardware in an audio block in the SAI .... 2479
Figure 767. Tristate strategy on SD output line on an inactive slot ....... 2480
Figure 768. Tristate on output data line in a protocol like I2S ........... 2481
Figure 769. Overrun detection error .............................................. 2482
Figure 770. FIFO underrun event .................................................. 2482
Figure 771. CAN subsystem .......................................................... 2518
Figure 772. FDCAN block diagram ............................................... 2520
Figure 773. Bit timing ................................................................. 2522
Figure 774. Transceiver delay measurement .................................. 2527
Figure 775. Pin control in Bus monitoring mode ......................... 2528
Figure 776. Pin control in Loop back mode ................................. 2530
Figure 777. Message RAM configuration ...................................... 2531
Figure 778. Standard Message ID filter path ................................. 2534
Figure 779. Extended Message ID filter path ................................ 2535
Figure 780. USB peripheral block diagram .................................. 2581
Figure 781. Packet buffer areas with examples of buffer description table locations .................................................. 2587
Figure 782. UCPD block diagram .................................................. 2628
Figure 783. Clock division and timing elements ............................. 2630
Figure 784. K-code transmission .................................................. 2632
Figure 785. Transmit order for various sizes of data ...................... 2633
Figure 786. Packet format ........................................................... 2634
Figure 787. Line format of Hard Reset .......................................... 2634
Figure 788. Line format of Cable Reset ........................................ 2635
Figure 789. BIST test data frame .................................................. 2636
List of figures

Figure 790. BIST Carrier Mode 2 frame ............................................................... 2636
Figure 791. UCPD BMC transmitter architecture .............................................. 2637
Figure 792. UCPD BMC receiver architecture .................................................. 2638
Figure 793. Ethernet high-level block diagram ................................................. 2671
Figure 794. DMA transmission flow (standard mode) ....................................... 2674
Figure 795. DMA transmission flow (OSP mode) ............................................. 2676
Figure 796. Receive DMA flow .......................................................................... 2678
Figure 797. Overview of MAC transmission flow .............................................. 2682
Figure 798. MAC reception flow ...................................................................... 2684
Figure 799. Packet filtering sequence ............................................................... 2688
Figure 800. Networked time synchronization .................................................... 2697
Figure 801. Propagation delay calculation in clocks supporting peer-to-peer path correction ........................................................... 2698
Figure 802. System time update using fine correction method .......................... 2708
Figure 803. TCP segmentation offload overview .............................................. 2725
Figure 804. TCP segmentation offload flow ..................................................... 2726
Figure 805. Header and payload fields of segmented packets ............................ 2729
Figure 806. Supported PHY interfaces ............................................................. 2739
Figure 807. SMA Interface block ...................................................................... 2739
Figure 808. MDIO packet structure (Clause 45) .............................................. 2740
Figure 809. MDIO packet structure (Clause 22) .............................................. 2741
Figure 810. SMA write operation flow ............................................................. 2742
Figure 811. Write data packet .......................................................................... 2743
Figure 812. Read data packet .......................................................................... 2743
Figure 813. Media independent interface (MII) signals ..................................... 2745
Figure 814. RMII block diagram ..................................................................... 2747
Figure 815. Transmission bit order ................................................................... 2748
Figure 816. Receive bit order ............................................................................ 2749
Figure 817. LPI transitions (Transmit, 100 Mbps) ............................................. 2757
Figure 818. LPI Tx clock gating (when LPITCSE = 1) ...................................... 2758
Figure 819. LPI transitions (receive, 100 Mbps) .............................................. 2759
Figure 820. Descriptor ring structure ............................................................... 2780
Figure 821. DMA descriptor ring ..................................................................... 2781
Figure 822. Transmit descriptor (read format) .................................................. 2782
Figure 823. Transmit descriptor write-back format .......................................... 2787
Figure 824. Transmit context descriptor format ............................................... 2791
Figure 825. Receive normal descriptor (read format) ........................................ 2795
Figure 826. Receive normal descriptor (write-back format) .............................. 2798
Figure 827. Receive context descriptor ............................................................ 2805
Figure 828. Generation of ETH_DMAISR flags ............................................. 2823
Figure 829. Block diagram of debug support infrastructure ............................ 2943
Figure 830. Product life cycle states and debug authentication ....................... 2947
Figure 831. JTAG TAP state machine .............................................................. 2950
Figure 832. CoreSight topology ....................................................................... 2972
Figure 833. Trace port interface unit (TPIU) ..................................................... 3055
Figure 834. Embedded cross trigger ............................................................... 3068
1 Documentation conventions

1.1 General information

The STM32H563/H573 and STM32H562 devices have an Arm® Cortex-M33 core.

1.2 List of abbreviations for registers

The following abbreviations are used in register descriptions:

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

---

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
b. This is an exhaustive list of all abbreviations applicable to STMicroelectronics microcontrollers, some of them may not be used in the current document.
1.3 Glossary

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

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

2.1 System architecture

The STM32H563/H573 and STM32H562 architecture relies on an Arm Cortex-M33 core optimized for execution thanks to an instruction cache having a direct access to the embedded Flash memory.

This architecture also features a 32-bit multilayer AHB bus matrix that interconnects:

- up to 13 masters:
  - Fast C-bus, connecting Cortex-M33 with TrustZone® mainline and FPU core C-bus to the internal SRAMs through the instruction cache
  - Slow C-bus, connecting Cortex-M33 with TrustZone mainline and FPU core C-bus to the external memories through the instruction cache
  - Cortex-M33 with TrustZone mainline and FPU core S-bus (three masters connected to three internal SRAMs without latency)
  - Cortex-M33 with TrustZone mainline and FPU core S-bus connected to the external memories through the data cache
  - GPDMA1 (general purpose DMA featuring two master ports)
  - GPDMA2 (general purpose DMA featuring two master ports)
  - SDMMC1
  - SDMMC2
  - Ethernet MAC

- up to 10 slaves:
  - internal Flash memory (2 Mbytes)
  - internal SRAM1 (256 Kbytes)
  - internal SRAM2 (64 Kbytes)
  - internal SRAM3 (320 Kbytes)
  - AHB1 peripherals and backup RAM (4-Kbyte BKPSRAM) including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
  - AHB2 peripherals
  - FMC (flexible memory controller)
  - OCTOSPI
  - AHB3 peripherals, including AHB to APB bridge and APB peripherals (connected to APB3)
  - AHB4 peripherals
The bus matrix provides access from a master to a slave, enabling concurrent access and efficient operation even when several high-speed peripherals work simultaneously. This architecture is shown in the figure below.

Figure 1. System architecture

![System architecture diagram]

2.1.1 Fast C-bus

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

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

2.1.2 Slow C-bus

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

2.1.3 S-bus

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

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

Note: *The Bus Matrix has a zero latency when accessing SRAM1, SRAM2 and SRAM3.*
2.1.4 **DCache S-bus**

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

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

2.1.5 **GPDMA1 and GPDMA2 buses**

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

2.1.6 **SDMMC1 and SDMMC2 controllers DMA buses**

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

2.1.7 **BusMatrix**

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

2.1.8 **AHB/APB bridges**

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

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

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

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

2.1.9 **Ethernet MAC**

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

The security architecture is based on Arm TrustZone with the Armv8-M mainline extension. The TrustZone security is activated by the TZEN option bit in the FLASH_OPTR register. When the TrustZone is enabled, the SAU (security-attribution unit) and IDAU (implementation-defined-attribution unit) defines the access permissions based on secure and non-secure states.

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

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

2.2.1 Default TrustZone security state

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

- **CPU**:
  - Cortex-M33 is in secure state after reset. The boot address must be at a secure address.
- **Memory map**:
  - SAU is fully secure after reset. Consequently, all memory map is fully secure. Up to height SAU configurable regions are available for security attribution.
- **Flash memory**:
  - The Flash memory security area is defined by watermark user options.
  - Flash block-based security attributions are non-secure after reset.
- **SRAMs**:
  - All SRAMs are secure after reset. MPCBBx (block-based memory protection controller) are secure.
- **External memories**:
  - FMC and OCTOSPI banks are secure after reset. MPCWMx (watermark-based memory protection controller) are secure.
- **Peripherals** (see *Table 1 and Table 2 for a list of securable and TrustZone-aware peripherals*)
  - Securable peripherals are non-secure after reset.
  - TrustZone-aware peripherals are non-secure after reset. Their secure configuration registers are secure.
- All GPIO are secure after reset.
• Interrupts:
  – NVIC: All interrupts are secure after reset. NVIC is banked for secure and non-secure state.
  – TZIC: All illegal access interrupts are disabled after reset (see GTZC TrustZone system architecture).

### 2.2.2 TrustZone peripheral classification

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

• Securable: peripheral protected by an AHB/APB firewall gate that is controlled from TZSC controller to define security properties

• TrustZone-aware: peripheral connected directly to AHB or APB bus and implementing a specific TrustZone behavior such as a subset of registers being secure.

Refer to GTZC TrustZone system architecture for more details.

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

<table>
<thead>
<tr>
<th>Table 1. Securable peripherals by TZSC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bus</strong></td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>AHB4</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>AHB2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>AHB1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Table 1. Securable peripherals by TZSC (continued)

<table>
<thead>
<tr>
<th>Bus</th>
<th>Peripheral</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VREFBUF</td>
</tr>
<tr>
<td></td>
<td>LPTIM1</td>
</tr>
<tr>
<td></td>
<td>LPTIM3</td>
</tr>
<tr>
<td></td>
<td>LPTIM4</td>
</tr>
<tr>
<td></td>
<td>LPTIM5</td>
</tr>
<tr>
<td></td>
<td>LPTIM6</td>
</tr>
<tr>
<td>APB3</td>
<td>I2C4</td>
</tr>
<tr>
<td></td>
<td>I2C3</td>
</tr>
<tr>
<td></td>
<td>LPUART1</td>
</tr>
<tr>
<td></td>
<td>SPI5</td>
</tr>
<tr>
<td></td>
<td>USB FS</td>
</tr>
<tr>
<td></td>
<td>SAI2</td>
</tr>
<tr>
<td></td>
<td>SAI1</td>
</tr>
<tr>
<td>APB2</td>
<td>SPI6</td>
</tr>
<tr>
<td></td>
<td>SPI4</td>
</tr>
<tr>
<td></td>
<td>TIM17</td>
</tr>
<tr>
<td></td>
<td>TIM16</td>
</tr>
<tr>
<td></td>
<td>TIM15</td>
</tr>
<tr>
<td></td>
<td>USART1</td>
</tr>
<tr>
<td></td>
<td>TIM8</td>
</tr>
<tr>
<td></td>
<td>SPI1</td>
</tr>
<tr>
<td></td>
<td>TIM1</td>
</tr>
</tbody>
</table>
Table 1. Securable peripherals by TZSC (continued)

<table>
<thead>
<tr>
<th>Bus</th>
<th>Peripheral</th>
</tr>
</thead>
<tbody>
<tr>
<td>APB1</td>
<td>UCPD</td>
</tr>
<tr>
<td></td>
<td>FDCAN2</td>
</tr>
<tr>
<td></td>
<td>FDCAN1</td>
</tr>
<tr>
<td></td>
<td>LPTIM2</td>
</tr>
<tr>
<td></td>
<td>DTS</td>
</tr>
<tr>
<td></td>
<td>UART12</td>
</tr>
<tr>
<td></td>
<td>UART9</td>
</tr>
<tr>
<td></td>
<td>UART8</td>
</tr>
<tr>
<td></td>
<td>UART7</td>
</tr>
<tr>
<td></td>
<td>HDMI-CEC</td>
</tr>
<tr>
<td></td>
<td>USART11</td>
</tr>
<tr>
<td></td>
<td>USART10</td>
</tr>
<tr>
<td></td>
<td>USART6</td>
</tr>
<tr>
<td></td>
<td>CRS</td>
</tr>
<tr>
<td></td>
<td>I3C1</td>
</tr>
<tr>
<td></td>
<td>I2C2</td>
</tr>
<tr>
<td></td>
<td>I2C1</td>
</tr>
<tr>
<td></td>
<td>UART5</td>
</tr>
<tr>
<td></td>
<td>UART4</td>
</tr>
<tr>
<td></td>
<td>USART3</td>
</tr>
<tr>
<td></td>
<td>USART2</td>
</tr>
<tr>
<td></td>
<td>SPI3/I2S3</td>
</tr>
<tr>
<td></td>
<td>SPI2/I2S2</td>
</tr>
<tr>
<td></td>
<td>IWDG</td>
</tr>
<tr>
<td></td>
<td>WWDG</td>
</tr>
<tr>
<td></td>
<td>TIM14</td>
</tr>
<tr>
<td></td>
<td>TIM13</td>
</tr>
<tr>
<td></td>
<td>TIM12</td>
</tr>
<tr>
<td></td>
<td>TIM7</td>
</tr>
<tr>
<td></td>
<td>TIM6</td>
</tr>
<tr>
<td></td>
<td>TIM5</td>
</tr>
<tr>
<td></td>
<td>TIM4</td>
</tr>
<tr>
<td></td>
<td>TIM3</td>
</tr>
<tr>
<td></td>
<td>TIM2</td>
</tr>
<tr>
<td>Bus</td>
<td>Peripheral</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>AHB3</td>
<td>EXTI</td>
</tr>
<tr>
<td></td>
<td>RCC</td>
</tr>
<tr>
<td></td>
<td>PWR</td>
</tr>
<tr>
<td>AHB4</td>
<td>OTFDEC1</td>
</tr>
<tr>
<td>AHB2</td>
<td>GPIOI</td>
</tr>
<tr>
<td></td>
<td>GPIOH</td>
</tr>
<tr>
<td></td>
<td>GPIOG</td>
</tr>
<tr>
<td></td>
<td>GPIOF</td>
</tr>
<tr>
<td></td>
<td>GPIOE</td>
</tr>
<tr>
<td></td>
<td>GPIOD</td>
</tr>
<tr>
<td></td>
<td>GPIOC</td>
</tr>
<tr>
<td></td>
<td>GPIOB</td>
</tr>
<tr>
<td></td>
<td>GPIOA</td>
</tr>
<tr>
<td>AHB1</td>
<td>GTZC</td>
</tr>
<tr>
<td></td>
<td>DCACHE</td>
</tr>
<tr>
<td></td>
<td>ICACHE</td>
</tr>
<tr>
<td></td>
<td>FLASH</td>
</tr>
<tr>
<td></td>
<td>GPDMA-2</td>
</tr>
<tr>
<td></td>
<td>GPDMA-1</td>
</tr>
<tr>
<td>APB3</td>
<td>TAMP</td>
</tr>
<tr>
<td></td>
<td>RTC</td>
</tr>
<tr>
<td></td>
<td>SBS</td>
</tr>
</tbody>
</table>
2.3 Memory organization

2.3.1 Introduction

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

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

Figure 2. Memory map based on IDAU mapping
All the memory map areas that are not allocated to on-chip memories and peripherals are considered “Reserved”. The following table gives the boundary addresses of the peripherals available in the devices.

### Table 3. Memory map and peripheral register boundary addresses

<table>
<thead>
<tr>
<th>Bus</th>
<th>Secure boundary address</th>
<th>Non-secure boundary address</th>
<th>Peripheral</th>
<th>Peripheral register map</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHB4</td>
<td>0x5700 1800 - 0x57FF FFFF</td>
<td>0x4700 1800 - 0x47FF FFFF</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5700 1400 - 0x5700 17FF</td>
<td>0x4700 1400 - 0x4700 17FF</td>
<td>OCTOSPI1</td>
<td>OCTOSPI register map</td>
</tr>
<tr>
<td></td>
<td>0x5700 0800 - 0x5700 13FF</td>
<td>0x4700 0800 - 0x4700 13FF</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5700 0400 - 0x5700 07FF</td>
<td>0x4700 0400 - 0x4700 07FF</td>
<td>FMC</td>
<td>FMC register map</td>
</tr>
<tr>
<td></td>
<td>0x5600 F800 - 0x5700 03FF</td>
<td>0x4600 F800 - 0x4700 03FF</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5600 F000 - 0x5600 F3FF</td>
<td>0x4600 F000 - 0x4600 F3FF</td>
<td>DLYBOS1</td>
<td>DLYB register map</td>
</tr>
<tr>
<td></td>
<td>0x5600 9000 - 0x5600 EFFF</td>
<td>0x4600 9000 - 0x4600 EFFF</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5600 8C00 - 0x5600 8FFF</td>
<td>0x4600 8C00 - 0x4600 8FFF</td>
<td>SDMMC2</td>
<td>SDMMC register map</td>
</tr>
<tr>
<td></td>
<td>0x5600 8800 - 0x5600 87FF</td>
<td>0x4600 8800 - 0x4600 87FF</td>
<td>DLYBSD2</td>
<td>DLYBSD register map</td>
</tr>
<tr>
<td></td>
<td>0x5600 8400 - 0x5600 83FF</td>
<td>0x4600 8400 - 0x4600 83FF</td>
<td>DLYBSD1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5600 8000 - 0x5600 83FF</td>
<td>0x4600 8000 - 0x4600 83FF</td>
<td>SDMMC1</td>
<td>SDMMC register map</td>
</tr>
<tr>
<td></td>
<td>0x5600 5400 - 0x5600 7FFF</td>
<td>0x4600 5400 - 0x4600 7FFF</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5600 5000 - 0x5600 53FF</td>
<td>0x4600 5000 - 0x4600 53FF</td>
<td>OTFDEC1</td>
<td>OTFDEC register map</td>
</tr>
<tr>
<td></td>
<td>0x5600 0000 - 0x5600 4FFF</td>
<td>0x4600 0000 - 0x4600 4FFF</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5402 4400 - 0x5402 43FF</td>
<td>0x4402 4400 - 0x4402 43FF</td>
<td>DEBUG</td>
<td>DEBUG register map</td>
</tr>
<tr>
<td></td>
<td>0x5402 3000 - 0x5402 3FF</td>
<td>0x4402 3000 - 0x4402 3FF</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5402 2000 - 0x5402 23FF</td>
<td>0x4402 2000 - 0x4402 23FF</td>
<td>EXTI</td>
<td>EXTI register map</td>
</tr>
<tr>
<td></td>
<td>0x5402 1000 - 0x5402 1FF</td>
<td>0x4402 1000 - 0x4402 1FF</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5402 0C00 - 0x5402 0FFF</td>
<td>0x4402 0C00 - 0x4402 0FFF</td>
<td>RCC</td>
<td>RCC register map</td>
</tr>
<tr>
<td></td>
<td>0x5402 0800 - 0x5402 0FFF</td>
<td>0x4402 0800 - 0x4402 0FFF</td>
<td>PWR</td>
<td>PWR register map</td>
</tr>
<tr>
<td></td>
<td>0x5402 0000 - 0x5402 07FF</td>
<td>0x4402 0000 - 0x4402 07FF</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5400 8000 - 0x5400 FFFF</td>
<td>0x4400 8000 - 0x4400 FFFF</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5400 7C00 - 0x5400 7FFF</td>
<td>0x4400 7C00 - 0x4400 7FFF</td>
<td>TAMP</td>
<td>TAMP register map</td>
</tr>
<tr>
<td></td>
<td>0x5400 7800 - 0x5400 78FF</td>
<td>0x4400 7800 - 0x4400 78FF</td>
<td>RTC</td>
<td>RTC register map</td>
</tr>
<tr>
<td></td>
<td>0x5400 7400 - 0x5400 77FF</td>
<td>0x4400 7400 - 0x4400 77FF</td>
<td>VREFBUF</td>
<td>VREFBUF register map</td>
</tr>
<tr>
<td></td>
<td>0x5400 5800 - 0x5400 73FF</td>
<td>0x4400 5800 - 0x4400 73FF</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>Bus</td>
<td>Secure boundary address</td>
<td>Non-secure boundary address</td>
<td>Peripheral</td>
<td>Peripheral register map</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------</td>
<td>-----------------------------</td>
<td>------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>APB2</td>
<td>0x520C 1000 - 0x520C 1FFF</td>
<td>0x420C 1000 - 0x420C 1FFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x520C 0C00 - 0x520C 0FFF</td>
<td>0x420C 0C00 - 0x420C 0FFF</td>
<td>SAES</td>
<td>SAES register map</td>
</tr>
<tr>
<td></td>
<td>0x520C 0800 - 0x520C 0BFF</td>
<td>0x420C 0800 - 0x420C 0BFF</td>
<td>RNG</td>
<td>RNG register map</td>
</tr>
<tr>
<td></td>
<td>0x520C 0400 - 0x520C 07FF</td>
<td>0x420C 0400 - 0x420C 07FF</td>
<td>HASH</td>
<td>HASH register map</td>
</tr>
<tr>
<td></td>
<td>0x520C 0000 - 0x520C 03FF</td>
<td>0x420C 0000 - 0x420C 03FF</td>
<td>AES</td>
<td>AES register map</td>
</tr>
<tr>
<td></td>
<td>0x520C C800 - 0x520C 03FF</td>
<td>0x420C C800 - 0x420C 03FF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x520C C400 - 0x520C C7FF</td>
<td>0x420C C400 - 0x420C C7FF</td>
<td>PSSI</td>
<td>PSSI register map</td>
</tr>
<tr>
<td></td>
<td>0x520C C000 - 0x520C C3FF</td>
<td>0x420C C000 - 0x420C C3FF</td>
<td>DCMI</td>
<td>DCMI register map</td>
</tr>
<tr>
<td></td>
<td>0x520C 8800 - 0x520C 8BFF</td>
<td>0x420C 8800 - 0x420C 8BFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x520C 8400 - 0x520C 87FF</td>
<td>0x420C 8400 - 0x420C 87FF</td>
<td>DAC1</td>
<td>DAC register map</td>
</tr>
<tr>
<td></td>
<td>0x520C 8000 - 0x520C 83FF</td>
<td>0x420C 8000 - 0x420C 83FF</td>
<td>ADC1 / ADC2</td>
<td>ADC register map</td>
</tr>
<tr>
<td></td>
<td>0x520C 2400 - 0x520C 7FFF</td>
<td>0x420C 2400 - 0x420C 7FFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bus</th>
<th>Secure boundary address</th>
<th>Non-secure boundary address</th>
<th>Peripheral</th>
<th>Peripheral register map</th>
</tr>
</thead>
<tbody>
<tr>
<td>APB3(Continued)</td>
<td>0x5400 0000 - 0x5400 03FF</td>
<td>0x4400 0000 - 0x4400 03FF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5400 0400 - 0x5400 07FF</td>
<td>0x4400 0400 - 0x4400 07FF</td>
<td>SBS</td>
<td>SBS register map</td>
</tr>
<tr>
<td></td>
<td>0x5400 0C00 - 0x5400 0FFF</td>
<td>0x4400 0C00 - 0x4400 0FFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5400 1000 - 0x5400 1FFF</td>
<td>0x4400 1000 - 0x4400 1FFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5400 1800 - 0x5400 1BFF</td>
<td>0x4400 1800 - 0x4400 1BFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5400 1C00 - 0x5400 1FFF</td>
<td>0x4400 1C00 - 0x4400 1FFF</td>
<td>LPTIM6</td>
<td>LPTIM register map</td>
</tr>
<tr>
<td></td>
<td>0x5400 2000 - 0x5400 23FF</td>
<td>0x4400 2000 - 0x4400 23FF</td>
<td>SPI5</td>
<td>SPI register map</td>
</tr>
<tr>
<td></td>
<td>0x5400 2C00 - 0x5400 2FFF</td>
<td>0x4400 2C00 - 0x4400 2FFF</td>
<td>I2C</td>
<td>I2C register map</td>
</tr>
<tr>
<td></td>
<td>0x5400 3000 - 0x5400 33FF</td>
<td>0x4400 3000 - 0x4400 33FF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5400 3C00 - 0x5400 3FFF</td>
<td>0x4400 3C00 - 0x4400 3FFF</td>
<td>LPTIM5</td>
<td>LPTIM register map</td>
</tr>
<tr>
<td></td>
<td>0x5400 4000 - 0x5400 43FF</td>
<td>0x4400 4000 - 0x4400 43FF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5400 4400 - 0x5400 47FF</td>
<td>0x4400 4400 - 0x4400 47FF</td>
<td>LPTIM4</td>
<td>LPTIM register map</td>
</tr>
<tr>
<td></td>
<td>0x5400 4800 - 0x5400 4BFF</td>
<td>0x4400 4800 - 0x4400 4BFF</td>
<td>LPTIM3</td>
<td>LPTIM register map</td>
</tr>
<tr>
<td></td>
<td>0x5400 5000 - 0x5400 53FF</td>
<td>0x4400 5000 - 0x4400 53FF</td>
<td>LPTIM2</td>
<td>LPTIM register map</td>
</tr>
<tr>
<td></td>
<td>0x5400 5400 - 0x5400 57FF</td>
<td>0x4400 5400 - 0x4400 57FF</td>
<td>LPTIM1</td>
<td>LPTIM register map</td>
</tr>
</tbody>
</table>

Table 3. Memory map and peripheral register boundary addresses (continued)
<table>
<thead>
<tr>
<th>Bus</th>
<th>Secure boundary address</th>
<th>Non-secure boundary address</th>
<th>Peripheral</th>
<th>Peripheral register map</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AHB2 (Continued)</strong></td>
<td>0x5202 2000 - 0x5202 23FF</td>
<td>0x4202 2000 - 0x4202 23FF</td>
<td>GPIOI</td>
<td>GPIO register map</td>
</tr>
<tr>
<td></td>
<td>0x5202 1C00 - 0x5202 1FFF</td>
<td>0x4202 1C00 - 0x4202 1FFF</td>
<td>GPIOH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5202 1800 - 0x5202 1BFF</td>
<td>0x4202 1800 - 0x4202 1BFF</td>
<td>GPIOG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5202 1400 - 0x5202 17FF</td>
<td>0x4202 1400 - 0x4202 17FF</td>
<td>GPIOF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5202 1000 - 0x5202 13FF</td>
<td>0x4202 1000 - 0x4202 13FF</td>
<td>GPIOE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5202 0C00 - 0x5202 0FFF</td>
<td>0x4202 0C00 - 0x4202 0FFF</td>
<td>GPIOD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5202 0800 - 0x5202 0BFF</td>
<td>0x4202 0800 - 0x4202 0BFF</td>
<td>GPIOC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5202 0400 - 0x5202 07FF</td>
<td>0x4202 0400 - 0x4202 07FF</td>
<td>GPIOB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5202 0000 - 0x5202 03FF</td>
<td>0x4202 0000 - 0x4202 03FF</td>
<td>GPIOA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5003 7400 - 0x51FF FFFF</td>
<td>0x4003 7400 - 0x41FF FFFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5003 6400 - 0x5003 73FF</td>
<td>0x4003 6400 - 0x4003 73FF</td>
<td>MPC_WM_BK</td>
<td>GTZC1 TZSC register map</td>
</tr>
<tr>
<td></td>
<td>0x5003 3800 - 0x5003 63FF</td>
<td>0x4003 3C00 - 0x4003 63FF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5003 2400 - 0x5003 37FF</td>
<td>0x4003 2400 - 0x4003 37FF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5003 1800 - 0x5003 23FF</td>
<td>0x4003 1800 - 0x4003 23FF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5003 1400 - 0x5003 17FF</td>
<td>0x4003 1400 - 0x4003 17FF</td>
<td>DCACHE</td>
<td>DCACHE register map</td>
</tr>
<tr>
<td></td>
<td>0x5003 0800 - 0x5003 13FF</td>
<td>0x4003 0800 - 0x4003 13FF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5003 0400 - 0x5003 07FF</td>
<td>0x4003 0400 - 0x4003 07FF</td>
<td>ICACHE</td>
<td>ICACHE register map</td>
</tr>
<tr>
<td></td>
<td>0x5002 9400 - 0x5003 03FF</td>
<td>0x4002 9400 - 0x4003 03FF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5002 8000 - 0x5002 93FF</td>
<td>0x4002 8000 - 0x4002 93FF</td>
<td>ETHERNET</td>
<td>ETHERNET register map</td>
</tr>
<tr>
<td></td>
<td>0x5002 7000 - 0x5002 7FFF</td>
<td>0x4002 7000 - 0x4002 7FFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5002 6000 - 0x5002 6FFF</td>
<td>0x4002 6000 - 0x4002 6FFF</td>
<td>RAMCFG</td>
<td>RAMCFG register map</td>
</tr>
<tr>
<td></td>
<td>0x5002 4000 - 0x5002 5FFF</td>
<td>0x4002 4000 - 0x4002 5FFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5002 3C00 - 0x5002 3FFF</td>
<td>0x4002 3C00 - 0x4002 3FFF</td>
<td>FMAC</td>
<td>FMAC register map</td>
</tr>
<tr>
<td></td>
<td>0x5002 3800 - 0x5002 3BFF</td>
<td>0x4002 3800 - 0x4002 3BFF</td>
<td>CORDIC</td>
<td>CORDIC register map</td>
</tr>
<tr>
<td></td>
<td>0x5002 3400 - 0x5002 37FF</td>
<td>0x4002 3400 - 0x4002 37FF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5002 3000 - 0x5002 33FF</td>
<td>0x4002 3000 - 0x4002 33FF</td>
<td>CRC</td>
<td>CRC register map</td>
</tr>
<tr>
<td></td>
<td>0x5002 2400 - 0x5002 2FFF</td>
<td>0x4002 2400 - 0x4002 2FFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5002 2000 - 0x5002 23FF</td>
<td>0x4002 2000 - 0x4002 23FF</td>
<td>FLASH</td>
<td>FLASH register map</td>
</tr>
<tr>
<td></td>
<td>0x5002 1000 - 0x5002 1FFF</td>
<td>0x4002 1000 - 0x4002 1FFF</td>
<td>DMA2</td>
<td>DMA2 register map</td>
</tr>
<tr>
<td></td>
<td>0x5002 0000 - 0x5002 0FFF</td>
<td>0x4002 0000 - 0x4002 0FFF</td>
<td>DMA1</td>
<td>DMA1 register map</td>
</tr>
</tbody>
</table>
### Table 3. Memory map and peripheral register boundary addresses (continued)

<table>
<thead>
<tr>
<th>Bus</th>
<th>Secure boundary address</th>
<th>Non-secure boundary address</th>
<th>Peripheral</th>
<th>Peripheral register map</th>
</tr>
</thead>
<tbody>
<tr>
<td>APB2</td>
<td>0x5001 6C00 - 0x5001 FFFF</td>
<td>0x4001 6C00 - 0x4001 FFFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5001 6400 - 0x5001 6BFF</td>
<td>0x4001 6400 - 0x4001 6BFF</td>
<td>USB_FS RAM</td>
<td>USB register map</td>
</tr>
<tr>
<td></td>
<td>0x5001 6000 - 0x5001 63FF</td>
<td>0x4001 6000 - 0x4001 63FF</td>
<td>USB_FS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5001 5C00 - 0x5001 5FFF</td>
<td>0x4001 5C00 - 0x4001 5FFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5001 5800 - 0x5001 5BFF</td>
<td>0x4001 5800 - 0x4001 5BFF</td>
<td>SAI2</td>
<td>SAI register map</td>
</tr>
<tr>
<td></td>
<td>0x5001 5400 - 0x5001 57FF</td>
<td>0x4001 5400 - 0x4001 57FF</td>
<td>SAI1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5001 5000 - 0x5001 53FF</td>
<td>0x4001 5000 - 0x4001 53FF</td>
<td>SPI6</td>
<td>SPI register map</td>
</tr>
<tr>
<td></td>
<td>0x5001 4C00 - 0x5001 4FFF</td>
<td>0x4001 4C00 - 0x4001 4FFF</td>
<td>SPI4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5001 4800 - 0x5001 4BFF</td>
<td>0x4001 4800 - 0x4001 4BFF</td>
<td>TIM17</td>
<td>TIM16/ TIM 17 register map</td>
</tr>
<tr>
<td></td>
<td>0x5001 4400 - 0x5001 47FF</td>
<td>0x4001 4400 - 0x4001 47FF</td>
<td>TIM16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5001 4000 - 0x5001 43FF</td>
<td>0x4001 4000 - 0x4001 43FF</td>
<td>TIM15</td>
<td>TIM15 register map</td>
</tr>
<tr>
<td></td>
<td>0x5001 3C00 - 0x5001 3FFF</td>
<td>0x4001 3C00 - 0x4001 3FFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5001 3800 - 0x5001 3BFF</td>
<td>0x4001 3800 - 0x4001 3BFF</td>
<td>USART1</td>
<td>USART register map</td>
</tr>
<tr>
<td></td>
<td>0x5001 3400 - 0x5001 37FF</td>
<td>0x4001 3400 - 0x4001 37FF</td>
<td>TIM8</td>
<td>TIMx register map</td>
</tr>
<tr>
<td></td>
<td>0x5001 3000 - 0x5001 33FF</td>
<td>0x4001 3000 - 0x4001 33FF</td>
<td>SPI1 / I2S1</td>
<td>SPI register map</td>
</tr>
<tr>
<td></td>
<td>0x5001 2C00 - 0x5001 2FFF</td>
<td>0x4001 2C00 - 0x4001 2FFF</td>
<td>TIM1</td>
<td>TIMx register map</td>
</tr>
<tr>
<td></td>
<td>0x5001 0000 - 0x5001 2BFF</td>
<td>0x4001 0000 - 0x4001 2BFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5000 E000 - 0x5000 FFFF</td>
<td>0x4000 E000 - 0x4000 FFFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5000 DC00 - 0x5000 DFFF</td>
<td>0x4000 DC00 - 0x4000 DFFF</td>
<td>UCPD1</td>
<td>UCPD register map</td>
</tr>
<tr>
<td></td>
<td>0x5000 B400 - 0x5000 DBFF</td>
<td>0x4000 B400 - 0x4000 DBFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5000 AC00 - 0x5000 B3FF</td>
<td>0x4000 AC00 - 0x4000 B3FF</td>
<td>FDCAN SRAM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5000 A800 - 0x5000 ABFF</td>
<td>0x4000 A800 - 0x4000 ABFF</td>
<td>FDCAN2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5000 A400 - 0x5000 A7FF</td>
<td>0x4000 A400 - 0x4000 A7FF</td>
<td>FDCAN1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5000 9800 - 0x5000 A3FF</td>
<td>0x4000 9800 - 0x4000 A3FF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5000 9400 - 0x5000 97FF</td>
<td>0x4000 9400 - 0x4000 97FF</td>
<td>LPTIM2</td>
<td>LPTIM register map</td>
</tr>
<tr>
<td></td>
<td>0x5000 9000 - 0x5000 93FF</td>
<td>0x4000 9000 - 0x4000 93FF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5000 8C00 - 0x5000 8FFF</td>
<td>0x4000 8C00 - 0x4000 8FFF</td>
<td>DTS</td>
<td>DTS register map</td>
</tr>
<tr>
<td></td>
<td>0x5000 8800 - 0x5000 8BFF</td>
<td>0x4000 8800 - 0x4000 8BFF</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5000 8400 - 0x5000 87FF</td>
<td>0x4000 8400 - 0x4000 87FF</td>
<td>UART12</td>
<td>UART register map</td>
</tr>
<tr>
<td></td>
<td>0x5000 8000 - 0x5000 83FF</td>
<td>0x4000 8000 - 0x4000 83FF</td>
<td>UART9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5000 7C00 - 0x5000 7FFF</td>
<td>0x4000 7C00 - 0x4000 7FFF</td>
<td>UART8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5000 7800 - 0x5000 7BFF</td>
<td>0x4000 7800 - 0x4000 7BFF</td>
<td>UART7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5000 7400 - 0x5000 77FF</td>
<td>0x4000 7400 - 0x4000 77FF</td>
<td>Reserved</td>
<td>-</td>
</tr>
</tbody>
</table>

**Bus**: APB1

**Memory map and peripheral register boundary addresses**

- **Secure boundary address**
- **Non-secure boundary address**
- **Peripheral**
- **Peripheral register map**
### Table 3. Memory map and peripheral register boundary addresses (continued)

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

The devices feature up to 644-Kbyte SRAMs:
- 256-Kbyte SRAM1
- 64-Kbyte SRAM2
- 320-Kbyte SRAM3
- 4-Kbyte BKPSRAM

These SRAMs can be accessed as bytes, half-words (16 bits) or full words (32 bits). These memories can be addressed both by CPU and DMAs.

The CPU can access the SRAM1, SRAM2 and SRAM3 through the system bus or through the C-bus depending on the selected address. The CPU can access the BKPSRAM through the system bus only.

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

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

2.3.4 Flash memory overview

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

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

2.3.5 Boot modes

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

If BOOT_ADDR is not yet configured, dedicated libraries (programmed during ST production), can be used for secure boot. They are located in system Flash memory:
- ST libraries in system Flash memory assist the application software boot with special features such as secure boot and secure firmware install (SFI-RSS).
- ST iROT (immutable root of trust) secure software in user Flash memory are used for secure firmware update and provisioning (SFU).

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

The embedded bootloader is located in the system memory, programmed by ST during production. Refer to the application note *STM32 microcontroller system memory boot mode* (AN2606).
3 **System security**

The STM32H563/H573 and STM32H562 are designed with a comprehensive set of security features, some of them based on the standard Arm TrustZone® technology.

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

This section explains the different security features available on the STM32H563/H573 and STM32H562 devices.

### 3.1 Key security features

- Resource isolation using privilege mode and Armv8-M mainline security extension of Cortex-M33, extended to securable I/Os, memories, and peripherals
- Secure firmware installation (SFI) with device unique cryptographic key pair
  - leveraging the on-chip immutable bootloader that supports the image download through USART, USB, I²C, I³C, SPI, FDCAN, and JTAG
- Boot entry: the platform makes it possible to select between native immutable root of trust or proprietary boot entry (in user flash memory).
- Security services (in system flash): the platform comes with native security services, embedded in the system memory to manage the root of trust services. Native root of trust services takes care of: platform security, including secure boot, secure updates of next boot level (uROT: updatable root of trust), secure debug control (debug reopening, regression control). The services can be personalized for each OEM, using the provisioning tools.
- Temporal isolation: boot levels are isolated thanks to HDPL (hide protect level) monotonic counter.
- Secure storage, featuring:
  - Five non volatile areas dedicated to secure storage, protected with HDPL and TrustZone.
  - Battery-powered volatile secure storage, automatically erased in case of tamper
  - Write-only key registers in the AES engines
  - Device 96-bit unique ID and JTAG 32-bit device-specific ID
  - Secure storage can rely upon SAES engine to encrypt the stored data, to benefit of DHUK properties. All data encrypted with the DHUK benefit of temporal and runtime isolation (HDPL and TrustZone), RHUK, EPOCH (version counter) properties.
    a) Data are isolated: all data encrypted/decrypted relying upon SAES+DHUK benefit of the variation of the DHUK for each different isolated area (HDPL0, HDPL1, HDPL2, HDPL3S, HDPL3NS).
    b) RHUK: hardware secret non volatile, unique per device keys (copy protection).
    c) EPOCH: counter incremented each time a regression is done (antirollback).
- General purpose cryptographic acceleration
  - AES 256-bit engine, supporting ECB, CBC, CTR, GCM, and CCM chaining modes
- Secure AES 256-bit security coprocessor, supporting ECB, CBC, CTR, GCM, and CCM chaining modes with side-channel counter-measures and mitigations
- HASH processor, supporting SHA-1 checksums and SHA-2 secure hash (SHA2, SHA2-384, SHA2-512)
- Public key accelerator (PKA) for RSA/DH (up to 4096 bits) and ECC (up to 640 bits), implementing side-channel counter measures and mitigations when manipulating secrets
- True random number generator (RNG), NIST SP800-90B precertified

- On-the-fly decryption of encrypted image stored on external flash memory connected through the OCTOSPI
  - Almost-zero latency with standard NOR flash memories
  - Can be used to encrypt the image using device-unique secret keys.
  - Automatic key-erase in case of tamper

- New flexible life-cycle scheme
  - Allows product maintenance in-the-field (debug reopening or regressions)
  - Allows protected firmware distribution in up to three steps: immutable root of trust, secure and non-secure application

- Active tamper and protection against temperature, voltage, and frequency attacks
  - Up to eight active inputs, eight active outputs tamper pins, available in different power modes (refer to TAMP low-power modes)

## 3.2 Secure install

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

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

The SFI native service leverages the following hardware security features:

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

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

## 3.3 Secure boot

Secure boot is an immutable code that is always executed after a system reset. As a root of trust, this code checks the device static protections and activates available device runtime protections, reducing the risk of invalid or malicious code running on the platform. As root of trust, the secure boot also checks the integrity and authenticity of the next level firmware before executing it.
The STM32H573 offer two options (using BOOT_UBE) to execute the immutable code after a reset:
- Security services when ST-iROT (code natively present in the system memory) is selected managing the secure boot of the next boot level. ST-iROT takes care of the next level integrity, authenticity using configuration done by OEMs. The next stage can be an ST updatable service (ST-uROT) or a proprietary one (uROT)
- Proprietary boot entry when the OEM manages the full chain (OEM-iROT). To be installed in user flash memory with proper security activation.

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

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

In the devices, the secure boot takes benefit of hardware security features such as:
- resource isolation using TrustZone (see Section 3.6)
- temporal isolation using hide protection levels (HDPL) (see Section 3.7.1)
- secure execution (see Section 3.8)
- secure install and update (see Section 3.2 and Section 3.4)
- per domain secure storage, with associated cryptographic engines if available (see Section 3.9 and Section 3.10)

This section describes the features specifically designed for secure boot.

### 3.3.1 Unique boot entry

Thanks to unique boot entry (BOOT_UBE) it is possible to select the boot entry point between security services in system flash (ST-iROT) and proprietary boot entry (OEM-iROT). This selection is possible only for products embedding cryptographic acceleration (STM32H573), when TrustZone is enabled (TZEN=0xB4).

When TrustZone is enabled (TZEN = 0xB4) and SECBOOT_LOCK secure option bit is cleared, the application selects a boot entry point located either in the system flash memory (see the next section), or in the secure user flash memory, at the address defined by SECBOOTADD option bytes.

When TrustZone is enabled (TZEN = 0xB4) and SECBOOT_LOCK secure option bit is set, the device unique boot entry is the unmodifiable secure address defined by SECBOOTADD option bytes. The application cannot modify these option bytes when SECBOOT_LOCK is set.

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

When TrustZone is disabled (TZEN = 0xC3) and NSBOOT_LOCK option bit is set, the device unique boot entry is the unmodifiable non-secure address defined by NSBOOTADD option bytes. The application cannot modify these option bytes when NSBOOT_LOCK is set.

*Note:* SECBOOT_LOCK and NSBOOT_LOCK can be changed only when PRODUCT_STATE = Open, Provisioning, or Regression.
For more information on the boot mechanisms, refer to Section 4: Boot modes.

### 3.3.2 Immutable root of trust in system flash memory

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

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

The STM32H563/H573 products offer security services:
- ST-DA (debug authentication), to manage the debug authentication control feature, allowing to control the debug reopening and regressions of the product for after sales (field return) of the product.

The STM32H573xx products offer security services:
- ST-iROT (immutable root of trust), to handle the secure provisioning, secure boot, secure updates of the first updatable level of the platform.

### 3.4 Secure update

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

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

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

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

### 3.5 Resource isolation using hide protect levels

In the STM32H563/H573 and STM32H562 devices, the hardware and software resources used to boot can be isolated. This is called temporal isolation.

It is based on a monotonic counter taking care of only incrementing the levels. When the counter is incremented, the resources of the previous levels become hidden (code and data).
3.6 Resource isolation using TrustZone

In the STM32H563/H573 and STM32H562 devices, the hardware and software resources can be partitioned so that they exist either in the secure world or in the non-secure world, as shown in Figure 3.

Figure 3. Secure/non-secure partitioning using TrustZone technology

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

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

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

3.6.1 TrustZone security architecture

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

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

Note: The TZEN option bit in the FLASH_OPTSR2_PRG register activates TrustZone security.
3.6.2 **Armv8-M security extension of Cortex-M33**

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

The Armv8-M TrustZone implementation in STM32H563/H573 and STM32H562 devices is composed of the following features:

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

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

3.6.3 **Memory and peripheral allocation using IDAU/SAU**

**Security attributes**

As illustrated on Figure 4, the Armv8-M non-secure memory view is similar to Armv7-M (found in Cortex-M4), with the difference that the secure memory is hidden. The secure memory view shows the flash memory, SRAM and peripherals that are only accessible while the Cortex processor executes in Secure state.

**Note:** Figure 4 shows the 32-bit address space viewed after SAU configuration by the secure code.
The Cortex processor state (and the associated rights) depends upon the security attribute assigned to the memory region where it is executed:

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

To manage transitions to the secure world, developers must create non-secure callable (NSC) regions containing valid entry points to the secure libraries. The first instruction in these entry points must be the new secure gate (SG) instruction, used by the non-secure code to call a secure function (see Figure 5).
Programming security attributes

In Cortex-M33, the static implementation defined attribution unit (IDAU) works in conjunction with the programmable security attribution unit (SAU) to assign a specific security attribute (S, NS, or NSC) to a specific address, as shown in Table 4.

<table>
<thead>
<tr>
<th>IDAU security attribution</th>
<th>SAU security attribution(1)</th>
<th>Final security attribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-secure</td>
<td>Secure</td>
<td>Secure</td>
</tr>
<tr>
<td></td>
<td>Secure-NSC</td>
<td>Secure-NSC</td>
</tr>
<tr>
<td></td>
<td>Non-secure</td>
<td>Non-secure</td>
</tr>
<tr>
<td>Secure-NSC</td>
<td>Secure</td>
<td>Secure</td>
</tr>
<tr>
<td></td>
<td>Non-secure</td>
<td>Secure-NSC</td>
</tr>
</tbody>
</table>

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

The Cortex-M33 can configure the SAU only in the secure-privilege state. When the TrustZone is enabled, the SAU defaults all addresses as secure (S). A secure boot application can then program the SAU to create NSC or NS regions, as shown in Table 4.

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

3.6.4 Memory and peripheral allocation using GTZC

Global TrustZone framework architecture

On top of the Armv8-M TrustZone security extension in Cortex-M33, the devices embed complementary security features that reinforce, in a flexible way, the isolation between secure and non-secure worlds. Unlike the SAU/IDAU, the GTZC can protect legacy memories and peripherals against non-secure transactions coming from other masters than the Cortex-M33.
When the TrustZone security is active, a peripheral is either securable through the GTZC, or is natively TrustZone-aware, as shown in the previous figure:

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

When a securable peripheral is made secure-only with the GTZC, if this peripheral is master on the interconnect, it automatically issues secure transactions. The SDMMC is an example of securable master. TrustZone-aware AHB masters like Cortex-M33 or DMAs, drive a secure signal in the AHB interconnect, according to their security mode, independently to the GTZC.
Note: Like with TrustZone, a peripheral can be made privileged-only with TZSC (see Section 3.7.2). In this case, if this peripheral is master on the interconnect, it automatically issues privileged transactions.

**Securing memories with TZSC and MPCBB**

The TZSC block in GTZC provides the capability to manage the security and privilege for all securable external memories, programming the MPCWM resources as defined in Table 5.

### Table 5. MPCWMx resources

<table>
<thead>
<tr>
<th>Memory</th>
<th>MPC resource</th>
<th>Type of filtering</th>
<th>Number of regions</th>
<th>Default security</th>
<th>On-the-fly decryption(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCTOSPI1</td>
<td>MPCWM1</td>
<td>Non-secure privileged or unprivileged region (watermarks)</td>
<td>2</td>
<td>Secure privileged(2)</td>
<td>Yes</td>
</tr>
<tr>
<td>FMC_NOR bank</td>
<td>MPCWM2</td>
<td></td>
<td>2</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>FMC_NAND / FMC_SDRAM bank</td>
<td>MPCWM3</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backup SRAM (BKPSRAM)/FMC_SDRAM</td>
<td>MPCWM4</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Using the OTFDEC.
2. Assuming TrustZone is activated on the device, non-secure unprivileged otherwise.

The MPCBB resources in GTZC provide the capability to configure the security and privilege of embedded SRAM blocks, as defined in Table 6.

### Table 6. MPCBBx resources

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

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

**Applying GTZC configurations**

The TZSC and MPCBB blocks can be used in one of the following ways:
- statically programmed during the secure boot, locked and not changed afterwards
- dynamically re-programmed using a specific application code or real-time kernel

When the dynamic option is selected and the configuration is not locked:
- MPCBB secure blocks or MPCWM non-secure region size can be changed by a secure software. This software must be privileged for MPCWM, can be unprivileged if the particular block is not privileged-only.
- The secure (respectively privilege) state of each peripheral can be changed writing to GTZC_TZSC_SECCFRGx (respectively GTZC_TZSC_PRIVCFGRx) registers.
Securing peripherals with TZSC

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

Note: When the TrustZone is deactivated, the resource isolation hardware GTZC can still be used to isolate peripherals to privileged code only (see Section 3.7.2). When the TrustZone is activated, peripherals are set as non-secure and unprivileged after reset.

TrustZone-aware peripherals

The devices include the following TrustZone-aware peripherals:
- GPIOA to GPIO1
- GTZCx_MPCBB, GTZCx_TZIC and GTZCx_TZSC (GTZC blocks)
- OTFDEC1, writable only in secure if TZEN = 1
- EXTI
- Flash memory
- RCC and PWR
- GPDMA1 and GPDMA2
- SBS registers
- RTC and TAMP
- MCU debug unit DBGMCU

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

For more details, refer to Section 3.6.5.

TrustZone illegal access controller (TZIC)

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

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

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

Note: By default, all events are masked.
3.6.5 Managing security in TrustZone-aware peripherals

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

**Embedded flash memory**

When the TrustZone security is enabled through option bytes (TZEN = 0xB4), the whole memory is secure after reset. The following protections, shown in Figure 7, are available to the application:

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

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

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

**Figure 7. Flash memory TrustZone protections**

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

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

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

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

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

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

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

Key registers in each OTFDEC are write-only.

See Section 3.10.3 for more details on this cryptographic engine.

**Direct memory access controllers (GPDMAx)**

When a DMA channel x is defined as secure (SECx = 1 in GPDMA_SECCFGR), the source and destination transfers can be independently set as secure or non-secure by a secure application using SSEC and DSEC bits in GPDMA_CxTR1. Table 7 summarizes these security options available in each DMA channel.

<table>
<thead>
<tr>
<th>Destination type</th>
<th>Secure DMA channel x (SECx = 1)</th>
<th>Non-secure DMA channel y (SECy = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure source</td>
<td>Secure source</td>
<td>Secure source</td>
</tr>
<tr>
<td>Non-secure source</td>
<td>OK(3)</td>
<td>Transfer blocked</td>
</tr>
</tbody>
</table>

1. When a transfer is blocked, the transfer completes but the corresponding writes are ignored, and reads return 0s. An illegal access event to TZIC is automatically triggered by the memory/peripheral used as source or destination.

2. If the source is a memory, the transfer is possible only if SSEC = 0, otherwise the transfer is blocked.

3. If the destination is a memory, the transfer is possible only if DSEC = 0, otherwise the transfer is blocked.

4. If the transfer is memory-to-memory, it is possible only if SSEC = 0 and DSEC = 0, otherwise it is blocked.

When a channel is configured as secure:

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

When switching to a non-secure state, the secure application must abort the channel or wait until the secure channel is completed before doing the switch.

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

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

**Power control (PWR)**

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

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

Other PWR configuration bits become secure:

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

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

**Secure clock and reset (RCC)**

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

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

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

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

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

See [Section 11: Reset and clock control (RCC)] for details.
Real time clock (RTC)

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

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

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

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

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

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

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

Tamper and backup registers (TAMP)

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

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

When TAMPSEC is set in TAMP_SECCFGR:

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

The application can also:

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

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

See Section 47: Tamper and backup registers (TAMP) for more details.
**General-purpose I/Os (GPIO)**

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

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

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

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

The TrustZone-aware logic around GPIO ports, used as alternate function, is summarized in Table 8.

When an analog function with an analog switch is used, the connection to the peripherals listed in Table 9 is blocked by hardware when the peripheral is non-secure and the I/O is secure.

Finally, regarding GPIO and security, Table 10 summarizes the list of I/Os without any hardware protection linked to TrustZone. The listed signals (input and/or outputs) are not blocked when the I/O is set as secure, and the associated peripheral is non-secure.

### Table 8. Secure alternate function between peripherals and allocated I/Os

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Allocated I/O pin</th>
<th>Input</th>
<th>Output</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure</td>
<td>Secure</td>
<td>I/O data</td>
<td>Peripheral data</td>
<td>-</td>
</tr>
<tr>
<td>Non-secure</td>
<td>Secure</td>
<td>0</td>
<td>0</td>
<td>Out of reset configuration</td>
</tr>
<tr>
<td>Secure</td>
<td>Non-secure</td>
<td>I/O data</td>
<td>Peripheral data</td>
<td>-</td>
</tr>
<tr>
<td>Non-secure</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 9. Non-secure peripheral functions that cannot be connected to secure I/Os

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

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Analog function(^{(1)})</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12</td>
<td>ADC12_INy (y = 1 to 17)</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

Finally, regarding GPIO and security, Table 10 summarizes the list of I/Os without any hardware protection linked to TrustZone. The listed signals (input and/or outputs) are not blocked when the I/O is set as secure, and the associated peripheral is non-secure.

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

Similarly, when a secure application sets PA0 as secure to be used as UART4_TX, if TAMP is non-secure, it can be programmed to capture the USART input traffic through the TAMP_IN signal.
It is important that, for each case described in Table 10, the secure application decides if a potential effect on data integrity or confidentiality is critical or not. For example, if the USART situation described above is not acceptable (data transiting on secure USART is confidential), then the secure application must configure the TAMP as secure even if not used by the secure application.

Table 10. Non-secure peripheral functions that can be connected to secure I/Os

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

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

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

**Extended interrupts and event controller (EXTI)**

When the TrustZone security is enabled (TZEN = 0xB4), the EXTI is able to protect event register bits from being modified by non-secure accesses. The protection can individually be activated per input event via the register bits in EXTI_SECCFGRx. When an input event is configured as secure, only a secure application can change the configuration (including security), change the masking or clear the status of this input event.

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

See Section 23: Extended interrupts and event controller (EXTI) for more details.

**System configuration, boot and security (SBS)**

Like all TrustZone-aware peripherals, when the TrustZone security is enabled (TZEN = 0xB4), a non-secure read/write access to a secured SBS register is RAZ/WI. Such access also generates an illegal access event that triggers a secure illegal access interrupt if the SBS illegal access event is not masked in the TZIC.

See Section 14: System configuration, boot, and security (SBS) for more details.
Microcontroller debug unit (DBGMCU)

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

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

The DBGCMU is a TrustZone-aware peripheral, managing accesses to its control registers as described in Table 11.

### Table 11. TrustZone-aware DBGMCU non-secure accesses management

<table>
<thead>
<tr>
<th>Debug profile</th>
<th>Peripheral status$^{(1)}$</th>
<th>DBG_xx_STOP control bits</th>
<th>Write access</th>
<th>Read access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-secure invasive (SPIDEN = 0)</td>
<td>NS</td>
<td>Yes (S$^{(2)}$ or NS)</td>
<td>None (S or NS)</td>
<td>Yes (S or NS)</td>
</tr>
<tr>
<td>Secure invasive (SPIDEN = 1)</td>
<td>NS</td>
<td>Yes (S or NS)</td>
<td>S</td>
<td>Yes (S only)</td>
</tr>
</tbody>
</table>

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

Refer to Section 75.12: Microcontroller debug unit (DBGMCU) for more details.

### 3.6.6 Activating TrustZone security

The TrustZone is deactivated by default in all STM32H563/H573 and STM32H562 devices. It can be activated by setting the TZEN option bit in FLASH_OPTSR2_PRG when PRODUCT_STATE is Open. Once TZEN has changed from disabled to enabled (0xC3 to 0xB4), the default security state, after reset, is always the following:

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

- **Embedded flash memory**
  - Flash memory nonvolatile secure areas (with their HDP zone), are defined with nonvolatile registers FLASH_SECWMxR (x = 1, 2). Default secure option bytes setup is all user flash secure, without HDP area defined.
  - Voltatile block-based security attributions of the flash memory are non-secure.

- **Embedded SRAMs**
  - All SRAMs are secure, as defined in GTZC/MPCBB (see Section 3.6.4). The secure boot code can change this security setup, making blocks secure or not.

- **External memories**
  - All memory devices connected to the FMC and OCTOSPI are secure, as defined in GTZC/MPCWM (see Section 3.6.4). The secure-boot code can change this security setup, making components secure or not.

- All GPIOs are secure.
All GPDMA channels are non-secure.
Backup registers are non-secure.
Peripherals and GTZC:
  - Securable peripherals are non-secure and unprivileged.
  - TrustZone-aware peripherals are non-secure, with their secure configuration registers being secure.
  - All illegal access interrupts in GTZC/TZIC are disabled.

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

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

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

After the TrustZone deactivation, most features mentioned in Section 3.6 are no longer available:
- The nonvolatile secure area of the embedded flash memory is deactivated, including the HDP area.
- Only the NVIC manages non-secure interrupts.
- All secure registers in TrustZone-aware peripherals are RAZ/WI.

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

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

3.7.1 Temporal isolation using secure hide protection (HDP)
When the TrustZone security is enabled (TZEN = 0xB4), the embedded flash memory allows to define an HDP area per watermarked-secure area of each bank (8-Kbyte page granularity). The code executed in this HDP area, with its related data and keys, can be hidden after boot, until the next system reset. The hide protection principle is shown in Figure 8.
Activation of HDP area in user flash memory is related to HDPL1: as soon as HDPL ≥ 2, data read, write, and instruction fetch (on the area defined by HDPx_PSTRT and HDPx_PEND in FLASH_HDPxR_PRG option bytes), are denied until the next device reset.

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

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

3.7.2 Resource isolation using Cortex privileged mode

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

Thanks to this hardware isolation technology, available even if TrustZone is disabled (TZEN = 0XC3), critical code or data can be protected against intentional or unintentional tampering from the more exposed unprivileged code.

Memory and peripheral privileged allocation using MPU

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

Memory and peripheral privileged allocation using GTZC

For the Cortex-M33 master, to complement the coarse isolation provided by the MPU, the GTZC reinforces, in a flexible way, the isolation between privileged and unprivileged tasks, for peripherals and selected memories.
For masters other than the Cortex-M33, the GTZC can assign them as unprivileged initiators, automatically protecting resources defined as privileged against this master.

- **Securing peripherals with TZSC (privileged-only)**
  In the devices, a peripheral is either securable privileged-only through GTZC, or is natively privileged-aware:
  - A securable privileged-only peripheral or memory is protected by an AHB/APB firewall gate controlled by the TZSC.
  - A privileged-aware peripheral or memory is connected directly to AHB or APB interconnect, implementing a specific behavior (for example, a subset of registers or a memory area is privilege-only).
  
  When such peripheral is made privileged-only with GTZC, if it is master on the interconnect (SDMMC), it automatically issues privileged transactions. Privilege-aware masters like GPDMA1 and GPDMA2, drive privileged signal in the AHB interconnect according to their internal privileged mode, independently to the GTZC.
  
  The list of securable peripherals can be found in *Section 5: Global TrustZone controller (GTZC).*

- **Securing memories with TZSC and MPCBB (privileged-only)**
  The TZSC logic in GTZC provides the capability to manage the privilege level for all securable external memories, programming the MPCWM resources defined in *Section 3.6.4.*
  
  Similarly, the TZSC logic in GTZC provides the capability to configure the privilege level of embedded SRAM blocks, programming the MPCBB resources defined in *Section 3.6.4.*

- **Error management (privileged-only)**
  - Any unprivileged transaction trying to access a privileged resource is considered as illegal. There is no illegal access event generated for illegal unprivileged read and write accesses.
  - The addressed resource follows a silent-fail behavior, returning all-0 data for read and ignoring any write.
  - When an illegal unprivileged access occurs, no bus error is generated, except when this is an instruction fetch, accessing a privileged memory or a peripheral register.
Managing security in privileged-aware peripherals

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

- **Embedded flash memory**
  
  By default, all registers can be read or programmed in both privileged and unprivileged modes.

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

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

  Regarding privileged protection, the devices offer the following features:
  
  - The system flash memory can be accessed both in privileged and unprivileged modes.
  
  - Each watermark-based secure area, including its secure HDP area, is accessible in secure-privileged and secure-unprivileged mode, if applicable.

  - Each 8-Kbyte page of the embedded flash memory can be programmed on-the-fly as privileged only, using the block-based privileged configuration registers FLASH_PRIV1BBRx and FLASH_PRIV2BBRx. An unprivileged page is accessible by privileged or unprivileged access.

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

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

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

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

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

- **Direct memory access controllers (GPDMAx)**

  When a DMA channel x is defined as privileged (PRIVx = 1 in GPDMA_PRIVCFGR), special rules (see Table 12) apply when accessing privileged/unprivileged source or destination.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Privileged DMA channel x (PRIVx = 1)</th>
<th>Unprivileged DMA channel y (PRIVy = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privileged</td>
<td>Privileged source</td>
<td>Unprivileged source</td>
</tr>
<tr>
<td>Unprivileged</td>
<td>OK</td>
<td>Transfer blocked(1)</td>
</tr>
</tbody>
</table>

1. When a transfer is blocked, the transfer completes but the corresponding writes are ignored, and reads return 0s.
• **Power control (PWR)**

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

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

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

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

• **Secure clock and reset (RCC)**

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

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

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

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

• **Real time clock (RTC)**

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

When PRIV bit is set in privileged-only RTC_PRIVCFGR:

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

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

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

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

• **Tamper and backup registers (TAMP)**

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

When PRIV bit is set in privileged-only TAMP_PRIVCFGR:

– Writing the TAMP registers is possible only in privileged mode, except for the backup registers and the monotonic counters that have their own protection setting.
– Reading the TAMP_SECCFGR or TAMP_PRIVCFGR is always possible in privilege and unprivileged modes. All the other TAMP registers can be read only in privilege mode, except for the backup registers and the monotonic counters that have their own protection setting.
The application can also:
– make TAMP_COUNT1R register read and write privileged-only by setting the CNTPRIV bit in TAMP_PRIVCFGR
– increase security for two of the three protection zones in backup registers, using BKPRWPRIV and BKPWPWPRIV bits in TAMP_PRIVCFGR:
  – Make protection zone 1 read privileged, write privileged.
  – Make protection zone 2 read privileged or unprivileged, write privileged.
  – Protection zone 3 is always read and write privileged or unprivileged.

- **General-purpose I/Os (GPIO)**
  All GPIO registers can be read and written by privileged and unprivileged accesses, whatever the security state (secure or non-secure).

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

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

  See Section 23: Extended interrupts and event controller (EXTI) for more details.

- **System configuration boot and security (SBS)**
  All SBS registers can be read and written in both privileged and unprivileged modes, except:
  – FPUSEC bit in SBS_SECCFGR registers (privileged only)
  – SBS registers for CPU configuration: SBS_CSLOCKR, SBS_FPUIMR, and SBS_CNSLCKR

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

### 3.8 Secure execution

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

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

#### 3.8.1 Memory protection unit (MPU)

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

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

The memory map and the programming of the non-secure and secure MPUs split memory into regions (up to eight for the non-secure, and up to twelve for the TrustZone). Secure MPU is only available when TrustZone is activated.
3.8.2 Embedded flash memory write protection

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

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

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

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

3.8.3 Tamper detection and response

Principle

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

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

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

Tamper detection sources

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

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

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

Note: Timestamps are automatically generated when a tamper event occurs.
The internal tamper sources are listed in Table 13.

### Table 13. Internal tampers in TAMP

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

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

### Response to tampers

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

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

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

If \(V_{\text{DD}}\) is not present, the secrets that are erased when \(V_{\text{DD}}\) is present, are only erased at the next \(V_{\text{DD}}\) power on.
Note: Device secret erase is also triggered by setting the BKERASE bit in TAMP_CR2, or by performing a PRODUCT_STATE regression as defined in Section 3.11.2.
Device secrets are not reset by system reset or when the device wakes up from Standby mode.

Software filtering mechanism

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

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

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

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

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

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

Tamper detection and low-power modes

The effect of low-power modes on a tamper detection are summarized in Table 14.

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

3.9 Secure storage

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

Figure 9 shows how key management service application can use the AES engine for example to compute external image decryption keys. Non-volatile keys are stored in the flash memory in a dedicated area including access control (see Section 3.7.1), while volatile
key storage consists in the battery-powered, tamper-protected SRAM or registers in TrustZone-aware TAMP.

*Figure 9* also shows keys that are manipulated by software (like OTFDEC keys), or keys that are managed only by hardware (like DHUK). More information on those hardware keys can be found in Section 3.9.1.

**Figure 9. Key management principle**

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

### 3.9.1 Hardware secret key management

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

There are three different sources of hardware secret keys:

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

- **BHK**: 256-bit application key stored in tamper-resistant volatile storage in TAMP. This key is written at boot time, then read/write locked to application until next reset.

- **XORK**: result of a XOR of BHK and DHUK
Those keys can be used in the following modes:

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

### 3.9.2 Unique ID

The devices store a 96-bit ID that is unique to each device (see Section 76.1: Unique device ID register (96 bits)). Application services can use this unique identity key to identify the product in the cloud network, or make it difficult for counterfeit devices or clones to inject untrusted data into the network.

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

### 3.10 Crypto engines

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

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

This is especially true for asymmetric cryptography.

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

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

### 3.10.1 Crypto engines features

Table 15 lists the accelerated cryptographic operations available in the devices. Two AES accelerators are available (both can be reserved to secure application only).

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

The devices provide an additional on-chip hardware AES encryption and decryption engine, that implements counter-measures and mitigations against power and electromagnetic side-channel attacks.

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

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

For more information, refer to Section 34: Secure AES coprocessor (SAES).
3.10.3 **On-the-fly decryption engine (OTFDEC)**

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

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

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

### 3.11 Product life-cycle

A typical IoT device life-cycle is summarized in Figure 10. For each step, the devices propose secure life-cycle management mechanisms embedded in the hardware.

**Figure 10. Device life-cycle security**
Additional details on the various phases and associated transitions, found either at the vendor or end-user premises, are summarized in Table 16.

**Table 16. Main product life-cycle transitions**

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

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

### 3.11.1 Product configurations and security services

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

Security services are provisioned by ST in system flash (immutable). They provides the root of trust of the platform managing the verification and the update of the first updatable code (uROT).
### 3.11.2 Life-cycle management

The product life-cycle allows to control access to different assets (code and data) of the product, including during development, manufacturing, and after sales.

It allows to provision the product with different distribution models taking care on the code and data provisioned.

#### Table 17. Typical product life-cycle phases

<table>
<thead>
<tr>
<th>PRODUCT_STATE + DebugState</th>
<th>Debug (default configuration)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Open</strong></td>
<td>Device open</td>
<td>Secure(^{(1)}) and non-secure</td>
</tr>
<tr>
<td><strong>Provisioning</strong></td>
<td>Debug partially opened (only non-secure)</td>
<td>HDPL3 + Non-secure only</td>
</tr>
<tr>
<td><strong>iROT-Provisioned</strong></td>
<td>Debug partially opened (only non-secure)</td>
<td>HDPL3 + Non-secure only</td>
</tr>
<tr>
<td><strong>TZ-Closed</strong></td>
<td>Debug partially opened (only non-secure)</td>
<td>HDPL3 + Non-secure only</td>
</tr>
<tr>
<td><strong>Closed</strong></td>
<td>Debug closed</td>
<td>NoDebug + debug authentication control</td>
</tr>
<tr>
<td><strong>Locked</strong></td>
<td>Debug locked</td>
<td>None</td>
</tr>
<tr>
<td><strong>NS-Regression</strong></td>
<td>Debug closed</td>
<td>NoDebug</td>
</tr>
<tr>
<td><strong>Regression</strong></td>
<td>Debug closed</td>
<td>NoDebug</td>
</tr>
<tr>
<td><strong>Debug Constrained</strong></td>
<td>Debug opened following debug authentication permissions</td>
<td>Depends upon permissions</td>
</tr>
</tbody>
</table>

---

1. Debug is not available when executing RSS code.
The supported transitions, summarized in Figure 11, can be requested (when available) through the debug interface or via the system bootloader.

Figure 11. PRODUCT_STATEs (simplified TrustZone activated view)
The full view of the life cycle includes the debug authentication part, to help product maintenance to manage field returns (in-the-field centers). The debug authentication control is ensured thanks to a protocol based on Arm PSA ADAC specification. It allows controlling that the host has a trusted certificate with permissions. Permissions definition allow a lot of flexibility, in our default model we consider: Full or Partial regression, or to open the debug for non-secure.

When TrustZone is disabled (TZEN = 0xC3), the state TZ-Closed does not exist, and the debug authentication is based on a password authentication method. For details on the debug authentication feature, refer to the corresponding application note.

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

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

This implies the following actions:

- set the PRODUCT_STATE in Closed state when the product has been configured
- When debug authentication control is based on certificates (when TrustZone enabled (TZEN = 0xB4)), then **provision the OEM-PublicKey and the SOC-Mask**. Setting the SOC-Mask to only propose the non-secure-debug and partial and full regression guaranty to be compliant with the Arm PSA security model.
- When debug authentication control is based on password (when TrustZone disabled (TZEN = 0xC3)), then only the HASH of the password has to be provisioned.

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

3.12 **Software intellectual property protection and collaborative development**

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

Collaborative development is summarized in *Figure 13*. Starting from a personalized device sold by STMicroelectronics, a vendor A can integrate a portion of hardware and software on a platform A, that can then be used by a vendor B, who does the same before deploying a final product to the end users.

*Note:* *Each platform vendor can provision individual platforms for development not to be connected to a production cloud network (“Development Platform X”).*
The features described hereafter contribute to securing the software intellectual property within such a collaborative development.
3.12.1 Software intellectual property protection

As described in Section 3.11.2, the hardware PRODUCT_STATE mechanism automatically controls the accesses to secrets provisioned in the device. The protection of these secrets are defined in Table 18.

Table 18. Software intellectual property protection with PRODUCT_STATE

<table>
<thead>
<tr>
<th>PRODUCT_STATE protection level</th>
<th>Secrets protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Device open</td>
</tr>
<tr>
<td>Provisioning</td>
<td>Debug partially opened (only non-secure)</td>
</tr>
<tr>
<td>iROT-Provisioned</td>
<td>Debug partially opened (only non-secure)</td>
</tr>
<tr>
<td>iROT-Provisioned</td>
<td>Debug partially opened (only non-secure)</td>
</tr>
<tr>
<td>TZ-Closed</td>
<td>Debug partially opened (only non-secure)</td>
</tr>
<tr>
<td>Closed</td>
<td>No debug</td>
</tr>
<tr>
<td>Locked</td>
<td>No debug</td>
</tr>
</tbody>
</table>

3.12.2 Software intellectual property protection with OTFDEC

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

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

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

Figure 14 illustrates this last case, with the provisioning, in a non-trusted environment, of software intellectual properties both in the embedded flash memory and in an external SPI flash memory (encrypted).

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

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

1. The user creates a SFI image, composed of:
   - encrypted internal firmware and data (including external flash memory drivers)
   - encrypted external firmware and data AES key (up to 4)
   - encrypted external firmware and data image

2. The secure bootloader stored in the system memory, loads the second part of the secure bootloader in SRAM2, through the supported communication ports (USART, SPI, I2C, FDCAN, USB, I3C and JTAG). This second part runs in the secure SRAM2 and is responsible for executing the SFI process, applying the SFI protocol thanks to the commands received through the above mentioned supported communication ports.

3. The internal flash memory is programmed with decrypted option bytes, internal firmware and data, and external firmware and data AES keys. Alternatively, device unique external firmware AES keys can be used instead of such global keys.

4. The OTFDEC is properly initialized with encrypted region information, including the corresponding external firmware and data AES key.

5. Running the SFI process, chunks of encrypted external firmware and data image are decrypted in the device, then re-encrypted in the OTFDEC.
6. After a chunk OTFDEC re-encryption, the user external flash memory programmer is responsible for programming the last encrypted chunks to the external SPI flash memories through the OCTOSPI.

**Secure boot**

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

1. Secure-boot firmware executes, programming the external firmware and data AES keys to the OTFDEC write-only key registers, along with the other needed information.

2. The application reads or executes the encrypted external firmware and data through the OCTOSPI in memory mapped mode, unless a tamper event is detected. In this case, all OTFDEC keys are erased and encrypted regions are read as 0 until the OTFDEC is properly initialized.

For more information on SFI solutions for the devices, refer to the AN4993 “Overview secure firmware install (SFI)”.

**3.12.3 Other software intellectual property protections**

The device additional protections to software intellectual property are:

- Invasive attacks such as physical tampering or perturbation are countered by detection then decommissioning of the device before the detected attack succeeds.

- Noninvasive attacks, such as side channel attacks, are countered by not leaking secret information via side channels (such as timing, power, and EM emissions).
4 Boot modes

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

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

Embedded bootloader

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

Refer to AN2606 “STM32 microcontroller system memory boot mode”.

Embedded root security services (RSS)

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

Refer to AN4992 “Overview secure firmware install (SFI)”.

Embedded immutable root of trust (ST-iROT)

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

Embedded debug authentication (ST-DA)

The embedded ST-DA in the system memory, programmed by ST during production. ST-DA is the library that manages the debug authentication protocol by allowing to securely reopen the debug or to launch regressions on secured products in the field.
4.1 STM32H562/H563 boot modes

Table 19 provides the detail of the boot mode when the TrustZone is disabled (TZEN = 0xC3), for the STM32H562/H563 devices.

Table 19. STM32H562/H563 Boot mode when TrustZone is disabled (TZEN = 0xC3)

<table>
<thead>
<tr>
<th>PRODUCT_STATE</th>
<th>BOOT0 pin</th>
<th>Boot address option-byte selection</th>
<th>Boot area</th>
<th>ST programmed default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>0</td>
<td>NSBOOTADD[31:8]</td>
<td>Boot address defined by user option byte NSBOOTADD[31:8]</td>
<td>Flash: 0x0800 0000</td>
</tr>
<tr>
<td>Provisioning</td>
<td>x</td>
<td>NA</td>
<td>RSS</td>
<td></td>
</tr>
<tr>
<td>Provisioned, Closed, Locked</td>
<td>x</td>
<td>NSBOOTADD[31:8]</td>
<td>Boot address defined by user option byte NSBOOTADD[31:8]</td>
<td>Flash: 0x0800 0000</td>
</tr>
</tbody>
</table>

Note: The BOOT_UBE is only available on STM32H573x devices. Refer to Section 4.2.

Table 20 provides the detail of the boot mode when the TrustZone is enabled (TZEN=0xB4), for the STM32H562/H563 devices.

Table 20. STM32H562/H563 Boot mode when TrustZone is enabled (TZEN = 0xB4)

<table>
<thead>
<tr>
<th>PRODUCT_STATE</th>
<th>BOOT0 pin</th>
<th>Boot address option-byte selection</th>
<th>Boot area</th>
<th>ST programmed default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>0</td>
<td>SECBOOTADD[31:8]</td>
<td>Boot address defined by user option byte SECBOOTADD[31:8]</td>
<td>Flash: 0x0C00 0000</td>
</tr>
<tr>
<td>Provisioning</td>
<td>x</td>
<td>NA</td>
<td>RSS</td>
<td></td>
</tr>
<tr>
<td>Provisioned, TZ_Closed, Closed, Locked</td>
<td>x</td>
<td>SECBOOTADD[31:8]</td>
<td>Boot address defined by user option byte SECBOOTADD[31:8]</td>
<td>Flash: 0x0C00 0000</td>
</tr>
</tbody>
</table>

Note: The BOOT_UBE is only available on STM32H573x devices. Refer to Section 4.2.

When TrustZone is enabled (TZEN=0xB4), the boot space must be in secure area. The SECBOOTADD[24:0] option bytes are used to select the boot secure memory address. A unique boot entry option can be selected by setting the SECBOOT_LOCK option bit.
4.2 STM32H573x boot modes

*Table 21* provides the detail of the boot mode when the TrustZone is disabled (TZEN = 0xC3), for the STM32H573x products.

<table>
<thead>
<tr>
<th>PRODUCT_STATE</th>
<th>BOOT0 pin</th>
<th>BOOT_UBE FLASH_OPSR [29:22]</th>
<th>Boot address option-byte selection</th>
<th>Boot area</th>
<th>ST programmed default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>0</td>
<td>NA</td>
<td>NSBOOTADD[31:8]</td>
<td>Boot address defined by user option byte NSBOOTADD[31:8]</td>
<td>Flash: 0x0800 0000</td>
</tr>
<tr>
<td>Provisioning</td>
<td>x</td>
<td>NA</td>
<td>NA</td>
<td>Bootloader</td>
<td>Bootloader</td>
</tr>
<tr>
<td>Provisioned, Closed, Locked</td>
<td>x</td>
<td>NA</td>
<td>NSBOOTADD[31:8]</td>
<td>Boot address defined by user option byte NSBOOTADD[31:8]</td>
<td>Flash: 0x0800 0000</td>
</tr>
</tbody>
</table>

*Table 22* provides the detail of the boot mode when the TrustZone is enabled (TZEN = 0xB4), for the STM32H573x products.

<table>
<thead>
<tr>
<th>PRODUCT_STATE</th>
<th>BOOT0 pin</th>
<th>BOOT_UBE FLASH_OPSR [29:22]</th>
<th>Boot address option-byte selection</th>
<th>Boot area</th>
<th>ST programmed default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>0</td>
<td>x</td>
<td>SECBOOTADD[31:8]</td>
<td>Boot address defined by user option byte SECBOOTADD[31:8]</td>
<td>Flash: 0x0C00 0000</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>0xB4</td>
<td>NA</td>
<td>Bootloader</td>
<td>Bootloader</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>0xC3</td>
<td>NA</td>
<td>ST-iROT</td>
<td>ST-iROT</td>
</tr>
<tr>
<td>Provisioning</td>
<td>x</td>
<td>NA</td>
<td>NA</td>
<td>RSS</td>
<td>RSS</td>
</tr>
<tr>
<td>Provisioned, TZ_Closed, Locked</td>
<td>x</td>
<td>0xC3</td>
<td>ST-iROT</td>
<td>ST-iROT</td>
<td>ST-iROT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0xB4</td>
<td>SECBOOTADD[31:8]</td>
<td>Boot address defined by user option byte SECBOOTADD[31:8]</td>
<td>Flash: 0x0C00 0000</td>
</tr>
</tbody>
</table>

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

5.1 Introduction

This section describes the global TrustZone controller (GTZC) block that contains the following sub-blocks:

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

- **MPCBB**: memory protection controller - block based
  This sub-block configures the internal RAM in a TrustZone-system product having segmented SRAM (pages of 512 bytes) with programmable-security and privileged attributes.

- **TZIC**: TrustZone illegal access controller
  This sub-block gathers all illegal access events in the system and generates a secure interrupt towards NVIC.

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

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

5.2 GTZC main features

The GTZC main features are listed below:

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

**GTZC TrustZone system architecture**

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

- a secure world, where usually security sensitive applications are run and critical resources are located
- a non-secure or public world (such as the usual non secure operating system and user space)
The TrustZone architecture is extended beyond AHB and Armv8-M with:

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

AHB and APB peripherals can be categorized as:

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

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

**Application information**

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

- programmed during secure boot only, locked and not changed afterwards
- dynamically re-programmed when using specific application code or secure kernel (microvisor). When not locked, MPC secure blocks or region size can be changed by secure software executing from the secure FLASH region or secure SRAM. Same remark applies to the GTZC1_TZSC_SECCFGRx/PRIVCFGRx registers that define secure/privileged state of each peripheral.
The Armv8-M security architecture with secure, securable, and TrustZone-aware peripherals is shown in Figure 15.

**Figure 15. GTZC in Armv8-M subsystem block diagram**

### 5.3 GTZC implementation

The STM32H563/H573 and STM32H562 devices embed one instance of GTZC.

<table>
<thead>
<tr>
<th>GTZC sub-blocks</th>
<th>GTZC1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TZSC</td>
<td>X</td>
</tr>
<tr>
<td>TZIC</td>
<td>X</td>
</tr>
<tr>
<td>MPCBB sub-block (number of MPCBB)</td>
<td>X (3)</td>
</tr>
</tbody>
</table>
Table 24 shows the address offset of GTZC sub-blocks versus GTZC base address (refer to Section 2.3 for GTZC1 base address).

<table>
<thead>
<tr>
<th>GTZC1 sub-block</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTZC1_TZSC</td>
<td>0x0</td>
</tr>
<tr>
<td>GTZC1_TZIC</td>
<td>0x400</td>
</tr>
<tr>
<td>GTZC1_MPCBB1</td>
<td>0x800</td>
</tr>
<tr>
<td>GTZC1_MPCBB2</td>
<td>0xC00</td>
</tr>
<tr>
<td>GTZC1_MPCBB3</td>
<td>0x1000</td>
</tr>
</tbody>
</table>

Table 25 describes the characteristics of the available MPCWM.

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

Table 26 describes the characteristics of the available MPCBB.

<table>
<thead>
<tr>
<th>GTZC</th>
<th>MPC</th>
<th>Resource</th>
<th>Memory size (Kbytes)</th>
<th>Block size (bytes)</th>
<th>Number of blocks</th>
<th>Number of super-blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MPCBB1</td>
<td>SRAM1</td>
<td>256</td>
<td>512</td>
<td>512</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>MPCBB2</td>
<td>SRAM2</td>
<td>64</td>
<td></td>
<td>128</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>MPCBB3</td>
<td>SRAM3</td>
<td>320</td>
<td></td>
<td>640</td>
<td>20</td>
</tr>
</tbody>
</table>

5.4 GTZC functional description

5.4.1 GTZC block diagram

Figure 16 describes the combined feature of TZSC, MPCBB, and TZIC. Each sub-block is controlled by its own AHB configuration port.

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

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

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

**Figure 16. GTZC block diagram**

5.4.2 Illegal access definition

Three different types of illegal access exist:

- **Illegal non-secure access**
  
  Any non-secure transaction trying to write a secure resource is considered as illegal, hence the addressed resource generates an illegal access interrupt for illegal write access and a bus error for illegal fetch access. There are some exceptions on secure and privileged configuration registers: these latter ones authorize non secure read access to secure registers (see GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx).
• Illegal secure access
  Any secure transaction trying to access non-secure block in internal block-based SRAM or watermarked memory is considered as illegal.
  A correct TZIC setting allows the capture of the associated event and then generates the GTZC_IRQn interrupt to the NVIC. This applies for read, write, and execute access.
  Concerning the MPCBB controller, there is an option to ignore secure data read/write access on non-secure SRAM blocks, by setting the SRWILADIS bit in the GTZC1_MPCBBz_CR register. Secure read and write data transactions are then allowed on non-secure SRAM blocks, while secure execution access remains not allowed.
  Any secure execute transaction trying to access a non-secure peripheral register is considered as illegal and generate a bus error.

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

5.4.3 TrustZone security controller (TZSC)

The TZSC is composed of a configurable set of registers, providing the following features:
• Control of secure and privileged state for all peripherals, done through:
  – GTZC1_TZSC_SECCFGRx registers to control AHB/APB firewall stubs for the securable peripherals
  – GTZC1_TZSC_PRIVCFGRx registers to control AHB/APB firewall stubs for the privileged peripherals

• For watermark memory protection controller (external memories and backup SRAM), two independent regions can be defined and the following fields are used to program:
  – the start of the first protected subregion on the external memory/backup SRAM: SUBA_START[10:0]
  – the length of the first protected subregion on the external memory/backup SRAM: SUBA_LENGTH[11:0]
  – the start of the second protected subregion on the external memory/backup SRAM: SUBB_START[10:0]
  – the length of the second protected subregion on the external memory/backup SRAM: SUBB_LENGTH[11:0]

A control register for each subregion can be used to enable/disable the watermark memory protection controller as well as defining the right attributes of each subregion.
In *Figure 17*, region x represents the external memory or backup SRAM region (such as FMC bank, OCTOSPI1, or BKPSRAM). Secure and privileged attributes of subregions A and B are independently configurable. When no subregions are defined or enabled on the region x, then the default attribute of the region x is set as secure-privileged.

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

**Table 27. Secure properties of subregions A and B**

<table>
<thead>
<tr>
<th>subregion A</th>
<th>subregion B</th>
<th>Properties of overlapped region A and B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-secure</td>
<td>Non-secure</td>
<td>Non-secure</td>
</tr>
<tr>
<td>Non-secure</td>
<td>Secure</td>
<td>Non-secure</td>
</tr>
<tr>
<td>Secure</td>
<td>Non-secure</td>
<td>Non-secure</td>
</tr>
<tr>
<td>Secure</td>
<td>Secure</td>
<td>Secure</td>
</tr>
</tbody>
</table>

**Table 28. Privileged properties of subregions A and B**

<table>
<thead>
<tr>
<th>subregion A</th>
<th>subregion B</th>
<th>Properties of overlapped region A and B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unprivileged</td>
<td>Unprivileged</td>
<td>Unprivileged</td>
</tr>
<tr>
<td>Unprivileged</td>
<td>Privileged</td>
<td>Unprivileged</td>
</tr>
<tr>
<td>Privileged</td>
<td>Unprivileged</td>
<td>Unprivileged</td>
</tr>
<tr>
<td>Privileged</td>
<td>Privileged</td>
<td>Privileged</td>
</tr>
</tbody>
</table>
5.4.4 Memory protection controller - block based (MPCBB)

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

![MPCBB block diagram](image)

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

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

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

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

5.4.5 TrustZone illegal access controller (TZIC)

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

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

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

5.4.6 Power-on/reset state

The power-on and reset state of the TZSC clear to 0 all bits of GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx, meaning that all securable peripherals are respectively set to non-secure and unprivileged.
For internal SRAMx (x = 1 to 3), all GTZC1_MPCBBz_SECCFGRx and GTZC1_MPCBBz_PRIVCFGRx are set:

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

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

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

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

### 5.5 GTZC interrupts

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

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit Sleep mode</th>
<th>Exit Stop mode</th>
<th>Exit Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTZC</td>
<td>Illegal access</td>
<td>All flags in GTZC1_TZIC_SRx</td>
<td>All bits in GTZC1_TZIC IERx</td>
<td>Write 1 in the bit GTZC1_TZIC_FCRx</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
5.6  GTZC1 TZSC registers

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

5.6.1  GTZC1 TZSC control register (GTZC1_TZSC_CR)

Address offset: 0x000
Reset value: 0x0000 0000
Secure privileged access only.

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

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

Bit 0  LCK: lock the configuration of GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx until next reset
This bit is cleared by default and once set, it can not be reset until system reset.
0: configuration of all GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx not locked
1: configuration of all GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx locked

5.6.2  GTZC1 TZSC secure configuration register 1 (GTZC1_TZSC_SECCFGR1)

Address offset: 0x010
Reset value: 0x0000 0000
Write-secure access only.

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

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

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

Bit 31  LPTIM2SEC: secure access mode for LPTIM2
0: non-secure
1: secure
Bit 30  **DTSSEC**: secure access mode for DTS
   0: non-secure
   1: secure

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

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

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

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

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

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

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

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

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

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

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

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

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

Bit 16  **UART5SEC**: secure access mode for UART5
   0: non-secure
   1: secure
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value 1</th>
<th>Value 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>UART4SEC: secure access mode for UART4</td>
<td>secure</td>
<td>non-secure</td>
</tr>
<tr>
<td>14</td>
<td>USART3SEC: secure access mode for USART3</td>
<td>secure</td>
<td>non-secure</td>
</tr>
<tr>
<td>13</td>
<td>USART2SEC: secure access mode for USART2</td>
<td>secure</td>
<td>non-secure</td>
</tr>
<tr>
<td>12</td>
<td>SPI3SEC: secure access mode for SPI3</td>
<td>secure</td>
<td>non-secure</td>
</tr>
<tr>
<td>11</td>
<td>SPI2SEC: secure access mode for SPI2</td>
<td>secure</td>
<td>non-secure</td>
</tr>
<tr>
<td>10</td>
<td>IWDGSEC: secure access mode for IWDG</td>
<td>secure</td>
<td>non-secure</td>
</tr>
<tr>
<td>9</td>
<td>WWDGSEC: secure access mode for WWDG</td>
<td>secure</td>
<td>non-secure</td>
</tr>
<tr>
<td>8</td>
<td>TIM14SEC: secure access mode for TIM14</td>
<td>secure</td>
<td>non-secure</td>
</tr>
<tr>
<td>7</td>
<td>TIM13SEC: secure access mode for TIM13</td>
<td>secure</td>
<td>non-secure</td>
</tr>
<tr>
<td>6</td>
<td>TIM12SEC: secure access mode for TIM12</td>
<td>secure</td>
<td>non-secure</td>
</tr>
<tr>
<td>5</td>
<td>TIM7SEC: secure access mode for TIM7</td>
<td>secure</td>
<td>non-secure</td>
</tr>
<tr>
<td>4</td>
<td>TIM6SEC: secure access mode for TIM6</td>
<td>secure</td>
<td>non-secure</td>
</tr>
<tr>
<td>3</td>
<td>TIM5SEC: secure access mode for TIM5</td>
<td>secure</td>
<td>non-secure</td>
</tr>
<tr>
<td>2</td>
<td>TIM4SEC: secure access mode for TIM4</td>
<td>secure</td>
<td>non-secure</td>
</tr>
<tr>
<td>1</td>
<td>TIM3SEC: secure access mode for TIM3</td>
<td>secure</td>
<td>non-secure</td>
</tr>
</tbody>
</table>
5.6.3 GTZC1 TZSC secure configuration register 2 (GTZC1_TZSC_SECCFGR2)

Address offset: 0x014
Reset value: 0x0000 0000

Write-secure access only.

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

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

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>LPTIM5SEC: secure access mode for LPTIM5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: non-secure</td>
</tr>
<tr>
<td></td>
<td>1: secure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>LPTIM4SEC: secure access mode for LPTIM4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: non-secure</td>
</tr>
<tr>
<td></td>
<td>1: secure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>LPTIM3SEC: secure access mode for LPTIM3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: non-secure</td>
</tr>
<tr>
<td></td>
<td>1: secure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>LPTIM1SEC: secure access mode for LPTIM1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: non-secure</td>
</tr>
<tr>
<td></td>
<td>1: secure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 27</th>
<th>I2C4SEC: secure access mode for I2C4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: non-secure</td>
</tr>
<tr>
<td></td>
<td>1: secure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 26</th>
<th>I2C3SEC: secure access mode for I2C3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: non-secure</td>
</tr>
<tr>
<td></td>
<td>1: secure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 25</th>
<th>LPUART1SEC: secure access mode for LPUART</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: non-secure</td>
</tr>
<tr>
<td></td>
<td>1: secure</td>
</tr>
</tbody>
</table>
Bit 24  **SPI5SEC**: secure access mode for SPI5  
   0: non-secure  
   1: secure  

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Bit 2  **UCPDSEC**: secure access mode for UCPD  
   0: non-secure  
   1: secure
5.6.4 GTZC1 TZSC secure configuration register 3 (GTZC1_TZSC_SECCFGR3)

Address offset: 0x018
Reset value: 0x0000 0000
Write-secure access only.

This register can be written only by secure privileged transaction when the corresponding GTZC1_TZSC_PRIVCFG is set to 1. If a given PRIV is not set, the equivalent SEC bit can be written by secure unprivileged transaction.

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

| Bit 31:27 Reserved, must be kept at reset value. |
| Bit 26 **RAMCFGSEC**: secure access mode for RAMSCFG       |
| 0: non-secure                                          |
| 1: secure                                             |
| Bit 25 Reserved, must be kept at reset value.          |
| Bit 24 **OCTOSPI1SEC**: secure access mode for OCTOSPI1 |
| 0: non-secure                                          |
| 1: secure                                             |
| Bit 23 **FMCSEC**: secure access mode for FMC           |
| 0: non-secure                                          |
| 1: secure                                             |
| Bit 22 **SDMMC2SEC**: secure access mode for SDMMC2     |
| 0: non-secure                                          |
| 1: secure                                             |
| Bit 21 **SDMMC1SEC**: secure access mode for SDMMC1     |
| 0: non-secure                                          |
| 1: secure                                             |
Bit 20 **PKASEC**: secure access mode for PKA
   0: non-secure
   1: secure

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

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

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

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

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

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

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

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

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

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

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

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

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

Bit 1 **VREFBUFSEC**: secure access mode for VREFBUF
   0: non-secure
   1: secure
**5.6.5 GTZC1 TZSC privilege configuration register 1 (GTZC1_TZSC_PRIVCFGR1)**

Address offset: 0x020

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by secure privileged transaction when the corresponding GTZC1_TZSC_SECFCFGR signal is set to 1. If a given SEC bit is not set, the equivalent PRIV bit can be read/written by non-secure privileged transaction.

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>LPTIM2_PRIV</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>DTSPRIV</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>UART12 PRIV</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>UART9 PRIV</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>UART7 PRIV</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>DAC1 PRIV</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>HDMICปรสิต</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>UART1 PRIV</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>UART0 PRIV</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>UART7 PRIV</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>UART6 PRIV</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>UART5 PRIV</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>UART4 PRIV</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>UART3 PRIV</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>UART2 PRIV</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>UART1 PRIV</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31 LPTIM2_PRIV: privileged access mode for LPTIM2

0: unprivileged
1: privileged

Bit 30 DTSPRIV: privileged access mode for DTS

0: unprivileged
1: privileged

Bit 29 UART12_PRIV: privileged access mode for UART12

0: unprivileged
1: privileged

Bit 28 UART9_PRIV: privileged access mode for UART9

0: unprivileged
1: privileged

Bit 27 UART8_PRIV: privileged access mode for UART8

0: unprivileged
1: privileged

Bit 26 UART7_PRIV: privileged access mode for UART7

0: unprivileged
1: privileged

Bit 25 DAC1_PRIV: privileged access mode for DAC1

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Bit 10  **IWDGPRIV**: privileged access mode for IWDG
  0: unprivileged
  1: privileged
Bit 9  **WWDGPRIV**: privileged access mode for WWDG
       0: unprivileged
       1: privileged

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

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

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

Bit 5  **TIM7PRIV**: privileged access mode for TIM7
       0: unprivileged
       1: privileged

Bit 4  **TIM6PRIV**: privileged access mode for TIM6
       0: unprivileged
       1: privileged

Bit 3  **TIM5PRIV**: privileged access mode for TIM5
       0: unprivileged
       1: privileged

Bit 2  **TIM4PRIV**: privileged access mode for TIM4
       0: unprivileged
       1: privileged

Bit 1  **TIM3PRIV**: privileged access mode for TIM3
       0: unprivileged
       1: privileged

Bit 0  **TIM2PRIV**: privileged access mode for TIM2
       0: unprivileged
       1: privileged

5.6.6  **GTZC1 TZSC privilege configuration register 2**
       (**GTZC1_TZSC_PRIVCFGR2**)
Address offset: 0x024

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by secure privileged transaction when the corresponding GTZC1_TZSC_SECCFGR signal is set to 1. If a given SEC bit is not set, the equivalent PRIV bit can be read/written by non-secure privileged transaction.

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

| Bit 31  | Bit 30  | Bit 29  | Bit 28  | Bit 27  | Bit 26  | Bit 25  | Bit 24  | Bit 23  | Bit 22  | Bit 21  | Bit 20  | Bit 19  | Bit 18  | Bit 17  | Bit 16  |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| LPTIM5PRIV: privileged access mode for LPTIM5 | LPTIM4PRIV: privileged access mode for LPTIM4 | LPTIM3PRIV: privileged access mode for LPTIM3 | LPTIM1PRIV: privileged access mode for LPTIM1 | I2C4PRIV: privileged access mode for I2C4 | I2C3PRIV: privileged access mode for I2C3 | LPUART1PRIV: privileged access mode for LPUART | SPI5PRIV: privileged access mode for SPI5 | SPI4PRIV: privileged access mode for SPI4 | TIM17PRIV: privileged access mode for TIM17 | TIM16PRIV: privileged access mode for TIM16 | TIM15PRIV: privileged access mode for TIM15 | USARTPRIV: privileged access mode for USART | USBBPRIV: privileged access mode for USB | SAF1PRIV: privileged access mode for SAFE | SAF2PRIV: privileged access mode for SAFE |
| rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       |

Bit 31 **LPTIM5PRIV**: privileged access mode for LPTIM5
0: unprivileged
1: privileged

Bit 30 **LPTIM4PRIV**: privileged access mode for LPTIM4
0: unprivileged
1: privileged

Bit 29 **LPTIM3PRIV**: privileged access mode for LPTIM3
0: unprivileged
1: privileged

Bit 28 **LPTIM1PRIV**: privileged access mode for LPTIM1
0: unprivileged
1: privileged

Bit 27 **I2C4PRIV**: privileged access mode for I2C4
0: unprivileged
1: privileged

Bit 26 **I2C3PRIV**: privileged access mode for I2C3
0: unprivileged
1: privileged

Bit 25 **LPUART1PRIV**: privileged access mode for LPUART
0: unprivileged
1: privileged

Bit 24 **SPI5PRIV**: privileged access mode for SPI5
0: unprivileged
1: privileged

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

Bit 19 **USBBPRIV**: privileged access mode for USB
0: unprivileged
1: privileged
Bit 18  **SAI2PRIV**: privileged access mode for SAI2
0: unprivileged
1: privileged

Bit 17  **SAI1PRIV**: privileged access mode for SAI1
0: unprivileged
1: privileged

Bit 16  **SPI6PRIV**: privileged access mode for SPI6
0: unprivileged
1: privileged

Bit 15  **SPI4PRIV**: privileged access mode for SPI4
0: unprivileged
1: privileged

Bit 14  **TIM17PRIV**: privileged access mode for TIM17
0: unprivileged
1: privileged

Bit 13  **TIM16PRIV**: privileged access mode for TIM16
0: unprivileged
1: privileged

Bit 12  **TIM15PRIV**: privileged access mode for TIM15
0: unprivileged
1: privileged

Bit 11  **USART1PRIV**: privileged access mode for USART1
0: unprivileged
1: privileged

Bit 10  **TIM8PRIV**: privileged access mode for TIM8
0: unprivileged
1: privileged

Bit 9  **SPI1PRIV**: privileged access mode for SPI1
0: unprivileged
1: privileged

Bit 8  **TIM1PRIV**: privileged access mode for TIM1
0: unprivileged
1: privileged

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

Bit 2  **UCPDPRI**: privileged access mode for UCPD
0: unprivileged
1: privileged

Bit 1  **FDCAN2PRIV**: privileged access mode for FDCAN2
0: unprivileged
1: privileged

Bit 0  **FDCAN1PRIV**: privileged access mode for FDCAN1
0: unprivileged
1: privileged
5.6.7 GTZC1 TZSC privilege configuration register 3
(GTZC1_TZSC_PRIVCFGR3)

Address offset: 0x028
Reset value: 0x0000 0000
Write-privileged access only.

This register can be read or written only by secure privileged transaction when the corresponding GTZC1_TZSC_SECCFGR signal is set to 1. If a given SEC bit is not set, the equivalent PRIV bit can be read/written by non-secure privileged transaction.

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

<table>
<thead>
<tr>
<th>Bit 31:27 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 26 RAMCFGPRIV: privileged access mode for RAMSCFG</td>
</tr>
<tr>
<td>0: unprivileged</td>
</tr>
<tr>
<td>1: privileged</td>
</tr>
<tr>
<td>Bit 25 Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 24 OCTOSPI1PRIV: privileged access mode for OCTOSPI1</td>
</tr>
<tr>
<td>0: unprivileged</td>
</tr>
<tr>
<td>1: privileged</td>
</tr>
<tr>
<td>Bit 23 FMCPRIV: privileged access mode for FMC</td>
</tr>
<tr>
<td>0: unprivileged</td>
</tr>
<tr>
<td>1: privileged</td>
</tr>
<tr>
<td>Bit 22 SDMMC2PRIV: privileged access mode for SDMMC2</td>
</tr>
<tr>
<td>0: unprivileged</td>
</tr>
<tr>
<td>1: privileged</td>
</tr>
<tr>
<td>Bit 21 SDMMC1PRIV: privileged access mode for SDMMC1</td>
</tr>
<tr>
<td>0: unprivileged</td>
</tr>
<tr>
<td>1: privileged</td>
</tr>
<tr>
<td>Bit 20 PKAPRIV: privileged access mode for PKA</td>
</tr>
<tr>
<td>0: unprivileged</td>
</tr>
<tr>
<td>1: privileged</td>
</tr>
<tr>
<td>Bit 19 SAESPRI: privileged access mode for SAES</td>
</tr>
<tr>
<td>0: unprivileged</td>
</tr>
<tr>
<td>1: privileged</td>
</tr>
</tbody>
</table>
Bit 18 **RNGPRIV**: privileged access mode for RNG
   0: unprivileged
   1: privileged

Bit 17 **HASHPRIV**: privileged access mode for HASH
   0: unprivileged
   1: privileged

Bit 16 **AESPRIV**: privileged access mode for AES
   0: unprivileged
   1: privileged

Bit 15 **DCMIPRIV**: privileged access mode for DCMI
   0: unprivileged
   1: privileged

Bit 14 **ADC12PRIV**: privileged access mode for ADC1 and ADC2
   0: unprivileged
   1: privileged

Bit 13 **DCACHEPRIV**: privileged access mode for DCACHE
   0: unprivileged
   1: privileged

Bit 12 **ICACHEPRIV**: privileged access mode for ICACHE
   0: unprivileged
   1: privileged

Bit 11 **ETHPRIV**: privileged access mode for register of ETH
   0: unprivileged
   1: privileged

Bit 10 **FMACPRIV**: privileged access mode for FMAC
   0: unprivileged
   1: privileged

Bit 9 **CORDICPRIV**: privileged access mode for CORDIC
   0: unprivileged
   1: privileged

Bit 8 **CRCPRIV**: privileged access mode for CRC
   0: unprivileged
   1: privileged

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

Bit 1 **VREFBUFPRIV**: privileged access mode for VREFBUF
   0: unprivileged
   1: privileged

Bit 0 **LPTIM6PRIV**: privileged access mode for LPTIM6
   0: unprivileged
   1: privileged
5.6.8 GTZC1 TZSC memory x subregion z watermark configuration register (GTZC1_TZSC_MPCWMxzCFGR) (z = A to B)

Address offset: Block A: 0x40 + 0x10 *(x - 1) (x = 1 to 4)
Address offset: Block B: 0x48 + 0x10 *(x - 1) (x = 1 to 4)
Reset value: 0x0000 0000
Secure privilege access only.

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

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

Bit 9 PRIV: Privileged subregion z of base region x
This bit is taken into account only if SREN is set.
0: Privileged and unprivileged accesses are granted in subregion z.
1: Only privileged accesses are granted in subregion z of region x.

Bit 8 SEC: Secure subregion z of base region x
This bit is taken into account only if SREN is set.
0: Only non-secure data accesses are granted to subregion z of region x.
1: Only secure data accesses are granted to subregion z of region x.

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

Bit 1 SRLOCK: subregion z lock
This bit, once set, can be cleared only by a system reset.
0: GTZC1_TZSC_MPCWMxCFGR, GTZC1_TZSC_MPCWMxAR and GTZC1_TZSC_MPCWMxBR can be written.
1: Writes to GTZC1_TZSC_MPCWMxCFGR, GTZC1_TZSC_MPCWMxAR and GTZC1_TZSC_MPCWMxBR are ignored.

Bit 0 SREN: subregion z enable
0: subregion z is disabled. Access control of base region x applies to any access between this subregion start- and end-addresses.
1: subregion z of region x is enabled. Access control defined in GTZC1_TZSC_MPCWMx_CFGR applies to any access between this subregion start- and end-addresses, both defined in GTZC1_TZSC_MPCWMxAR and GTZC1_TZSC_MPCWMxBR.

Note: External memories that are watermark controlled start fully non-secure/unprivileged at reset when TZEN = 0xC3. When TZEN = 0xB4, external memories start fully secure/fully privileged (inverted reset-value).
5.6.9  GTZC1 TZSC memory x subregion A watermark register
(GTZC1_TZSC_MPCWMxAR)

Address offset: 0x44 + 0x10 *(x - 1) (x = 1 to 4)
Reset value: 0x0000 0000

The given reset value is valid when TZEN = 0xB4. The reset value is 0x0800 0000 when
TZEN = 0xC3.

Secure privilege access only.

When SUBA_START + SUBA_LENGTH is higher than the maximum size allowed for the
memory, a saturation of SUBA_LENGTH is applied automatically.

When an overlap of subregion A and B exists, secure/privileged attributes of both
subregions apply on the common section (see Section 5.4.3)

| Bits 31:28 Reserved, must be kept at reset value. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits 27:16 SUBA_LENGTH[11:0]: Length of subregion A in region x |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 |  1 |  0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits 15:11 Reserved, must be kept at reset value. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 10:0 SUBA_START[10:0]: Start of subregion A in region x

This field defines the address offset of the subregion A, to be multiplied by the granularity defined in Table 25.

External memories that are watermark controlled, start fully non-secure at reset when
TZEN = 0xC3. When TZEN = 0xB4, external memories start fully secure (inverted reset
value).
5.6.10 GTZC1 TZSC memory x subregion B watermark register (GTZC1_TZSC_MPCWMxBR)

Address offset: 0x4C + 0x10 *(x - 1) (x = 1 to 4)

Reset value: 0x0000 0000

The given reset value is valid when TZEN = 0xB4. The reset value is 0x0800 0000 when TZEN = 0xC3.

Secure privilege access only.

When SUBB_START + SUBB_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBB_LENGTH is applied automatically.

When an overlap of subregion A and B exists, secure/privileged attributes of both subregions apply on the common section (see Section 5.4.3)

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 SUBB_LENGTH[11:0]: Length of subregion B in region x
This field defines the length of the subregion B, to be multiplied by the granularity defined in Table 25.
When SUBB_START + SUBB_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBB_LENGTH is applied automatically.
If SUBB_LENGTH = 0, the subregion B is disabled.(SREN bit in GTZC1_TZSC_MPCWMxBCFG is cleared).

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 SUBB_START[10:0]: Start of subregion B in region x
This field defines the address offset of the subregion B, to be multiplied by the granularity defined in Table 25, versus the start of the region x.
External memories that are watermark controlled, start fully non-secure at reset when TZEN = 0xC3. When TZEN = 0xB4, external memories start fully secure (inverted reset value).
## 5.6.11 GTZC1 TZSC register map

**Table 30. GTZC1 TZSC register map and reset values**

| Offset | Register                  | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x004-  | Reserved                  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x01C  | Reserved                  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x02C-  | Reserved                  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
### Table 30. GTZC1 TZSC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>SUBB_LENGTH[11:0]</th>
<th>SUBB_START[10:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x04C</td>
<td>GTZC1_TZSC_MPCWM1BR</td>
<td>0000000000000000</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x050</td>
<td>GTZC1_TZSC_MPCWM2ACFGR</td>
<td>0000000000000000</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x054</td>
<td>GTZC1_TZSC_MPCWM2BR</td>
<td>0000000000000000</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x058</td>
<td>GTZC1_TZSC_MPCWM2ACFGR</td>
<td>0000000000000000</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x05C</td>
<td>GTZC1_TZSC_MPCWM2BR</td>
<td>0000000000000000</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x060</td>
<td>GTZC1_TZSC_MPCWM3ACFGR</td>
<td>0000000000000000</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x064</td>
<td>GTZC1_TZSC_MPCWM3BR</td>
<td>0000000000000000</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x068</td>
<td>GTZC1_TZSC_MPCWM3ACFGR</td>
<td>0000000000000000</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x06C</td>
<td>GTZC1_TZSC_MPCWM3BR</td>
<td>0000000000000000</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x070</td>
<td>GTZC1_TZSC_MPCWM4ACFGR</td>
<td>0000000000000000</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x074</td>
<td>GTZC1_TZSC_MPCWM4BR</td>
<td>0000000000000000</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x078</td>
<td>GTZC1_TZSC_MPCWM4ACFGR</td>
<td>0000000000000000</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>0x07C</td>
<td>GTZC1_TZSC_MPCWM4BR</td>
<td>0000000000000000</td>
<td>0000000000000000</td>
</tr>
</tbody>
</table>

Refer to Table 24: GTZC1 sub-block address offset.
5.7 GTZC1 TZIC registers

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

5.7.1 GTZC1 TZIC interrupt enable register 1 (GTZC1_TZIC_IER1)

Address offset: 0x000
Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>LPTIM2IE: illegal access interrupt enable for LPTIM2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>DTSIE: illegal access interrupt enable for DTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>UART12IE: illegal access interrupt enable for UART12</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>UART9IE: illegal access interrupt enable for UART9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 27</th>
<th>UART8IE: illegal access interrupt enable for UART8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 26</th>
<th>UART7IE: illegal access interrupt enable for UART7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 25</th>
<th>DAC1IE: illegal access interrupt enable for DAC1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 24</th>
<th>HDMICECIE: illegal access interrupt enable for HDMICEC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>LPTIM2IE</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>DTSIE</td>
</tr>
<tr>
<td>29</td>
<td>UART12IE</td>
</tr>
<tr>
<td>28</td>
<td>UART9IE</td>
</tr>
<tr>
<td>27</td>
<td>UART8IE</td>
</tr>
<tr>
<td>26</td>
<td>UART7IE</td>
</tr>
<tr>
<td>25</td>
<td>DAC1IE</td>
</tr>
<tr>
<td>24</td>
<td>HDMICECIE</td>
</tr>
<tr>
<td>Bit 23</td>
<td>USART11IE: illegal access interrupt enable for USART11</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 22</th>
<th>USART10IE: illegal access interrupt enable for USART10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 21</th>
<th>USART6IE: illegal access interrupt enable for USART6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 20</th>
<th>CRSIE: illegal access interrupt enable for CRS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 19</th>
<th>I3C1IE: illegal access interrupt enable for I3C1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 18</th>
<th>I2C2IE: illegal access interrupt enable for I2C2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 17</th>
<th>I2C1IE: illegal access interrupt enable for I2C1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 16</th>
<th>UART5IE: illegal access interrupt enable for UART5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>UART4IE: illegal access interrupt enable for UART4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 14</th>
<th>USART3IE: illegal access interrupt enable for USART3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 13</th>
<th>USART2IE: illegal access interrupt enable for USART2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 12</th>
<th>SPI3IE: illegal access interrupt enable for SPI3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 11</th>
<th>SPI2IE: illegal access interrupt enable for SPI2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 10</th>
<th>IWDGIIE: illegal access interrupt enable for IWDG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 9</th>
<th>WWDGIE: illegal access interrupt enable for WWDG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>
Bit 8  **TIM14IE**: illegal access interrupt enable for TIM14  
0: interrupt disabled  
1: interrupt enabled  

Bit 7  **TIM13IE**: illegal access interrupt enable for TIM13  
0: interrupt disabled  
1: interrupt enabled  

Bit 6  **TIM12IE**: illegal access interrupt enable for TIM12  
0: interrupt disabled  
1: interrupt enabled  

Bit 5  **TIM7IE**: illegal access interrupt enable for TIM7  
0: interrupt disabled  
1: interrupt enabled  

Bit 4  **TIM6IE**: illegal access interrupt enable for TIM6  
0: interrupt disabled  
1: interrupt enabled  

Bit 3  **TIM5IE**: illegal access interrupt enable for TIM5  
0: interrupt disabled  
1: interrupt enabled  

Bit 2  **TIM4IE**: illegal access interrupt enable for TIM4  
0: interrupt disabled  
1: interrupt enabled  

Bit 1  **TIM3IE**: illegal access interrupt enable for TIM3  
0: interrupt disabled  
1: interrupt enabled  

Bit 0  **TIM2IE**: illegal access interrupt enable for TIM2  
0: interrupt disabled  
1: interrupt enabled  

### 5.7.2 GTZC1 TZIC interrupt enable register 2 (GTZC1_TZIC_IER2)

Address offset: 0x004  
Reset value: 0x0000 0000  
Secure privileged access only.  
This register is used to enable interrupt of illegal access.
Bit 31  **LPTIM5IE**: illegal access interrupt enable for LPTIM5
0: interrupt disabled
1: interrupt enabled

Bit 30  **LPTIM4IE**: illegal access interrupt enable for LPTIM4
0: interrupt disabled
1: interrupt enabled

Bit 29  **LPTIM3IE**: illegal access interrupt enable for LPTIM3
0: interrupt disabled
1: interrupt enabled

Bit 28  **LPTIM1IE**: illegal access interrupt enable for LPTIM1
0: interrupt disabled
1: interrupt enabled

Bit 27  **I2C4IE**: illegal access interrupt enable for I2C4
0: interrupt disabled
1: interrupt enabled

Bit 26  **I2C3IE**: illegal access interrupt enable for I2C3
0: interrupt disabled
1: interrupt enabled

Bit 25  **LPUART11IE**: illegal access interrupt enable for LPUART
0: interrupt disabled
1: interrupt enabled

Bit 24  **SPI5IE**: illegal access interrupt enable for SPI5
0: interrupt disabled
1: interrupt enabled

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

Bit 19  **USBIE**: illegal access interrupt enable for USB
0: interrupt disabled
1: interrupt enabled

Bit 18  **SAI2IE**: illegal access interrupt enable for SAI2
0: interrupt disabled
1: interrupt enabled

Bit 17  **SAI1IE**: illegal access interrupt enable for SAI1
0: interrupt disabled
1: interrupt enabled

Bit 16  **SPI6IE**: illegal access interrupt enable for SPI6
0: interrupt disabled
1: interrupt enabled

Bit 15  **SPI4IE**: illegal access interrupt enable for SPI4
0: interrupt disabled
1: interrupt enabled

Bit 14  **TIM17IE**: illegal access interrupt enable for TIM17
0: interrupt disabled
1: interrupt enabled
5.7.3 GTZC1 TZIC interrupt enable register 3 (GTZC1_TZIC_IER3)

Address offset: 0x008
Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.
Bits 31:27Reserved, must be kept at reset value.

<table>
<thead>
<tr>
<th>Bit 26</th>
<th>RAMCFGIE: illegal access interrupt enable for RAMSCFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

Bit 25Reserved, must be kept at reset value.

<table>
<thead>
<tr>
<th>Bit 24</th>
<th>OCTOSPI1IE: illegal access interrupt enable for OCTOSPI1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 23</th>
<th>FMCIE: illegal access interrupt enable for FMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 22</th>
<th>SDMMC2IE: illegal access interrupt enable for SDMMC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 21</th>
<th>SDMMC1IE: illegal access interrupt enable for SDMMC1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 20</th>
<th>PKAIE: illegal access interrupt enable for PKA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 19</th>
<th>SAESIE: illegal access interrupt enable for SAES</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 18</th>
<th>RNGIE: illegal access interrupt enable for RNG</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 17</th>
<th>HASHIE: illegal access interrupt enable for HASH</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 16</th>
<th>AESIE: illegal access interrupt enable for AES</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>DCMIE: illegal access interrupt enable for DCMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 14</th>
<th>ADC12IE: illegal access interrupt enable for ADC1 and ADC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 13</th>
<th>DCACHEIE: illegal access interrupt enable for DCACHE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 12</th>
<th>ICACHEIE: illegal access interrupt enable for ICACHE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>
5.7.4 GTZC1 TZIC interrupt enable register 4 (GTZC1_TZIC_IER4)

Address offset: 0x00C

Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>Res.</td>
<td>MPCB</td>
<td>SRAM3</td>
<td>MPCB</td>
<td>SRAM2</td>
<td>MPCB</td>
<td>SRAM1</td>
<td>Res.</td>
<td>Res.</td>
<td>BKPSR</td>
<td>FMC</td>
<td>OCTOS</td>
<td>TZC11</td>
<td>TZSC11</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>B3_RE</td>
<td>IE</td>
<td>B2_RE</td>
<td>IE</td>
<td>B1_RE</td>
<td>IE</td>
<td></td>
<td></td>
<td>AMIE</td>
<td>MEMIE</td>
<td>P11_MEMIE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 MPCBB3_REGIE: illegal access interrupt enable for MPCBB3 registers
0: interrupt disabled
1: interrupt enabled

Bit 28 SRAM3IE: illegal access interrupt enable for SRAM3
0: interrupt disabled
1: interrupt enabled
<table>
<thead>
<tr>
<th>Bit 27</th>
<th>MPCBB2_REGIE: illegal access interrupt enable for MPCBB2 registers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 26</td>
<td>SRAM2IE: illegal access interrupt enable for SRAM2</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 25</td>
<td>MPCBB1_REGIE: illegal access interrupt enable for MPCBB1 registers</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 24</td>
<td>SRAM1IE: illegal access interrupt enable for SRAM1</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td></td>
<td>Bits 23:21 Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 20</td>
<td>BKPSRAMIE: illegal access interrupt enable for MPCWM4 (BKPSRAM) memory bank</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 19</td>
<td>FMC_MEMIE: illegal access interrupt enable for MPCWM2 (FMC_NOR bank), MPCWM3 (FMC_NAND bank and FMC_SDRAM bank 1), and MPCWM4 (FMC_SDRAM bank 2)</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 18</td>
<td>OCTOSPI1_MEMIE: illegal access interrupt enable for MPCWM1 (OCTOSPI1) memory bank</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 17</td>
<td>TZIC1IE: illegal access interrupt enable for GTZC1 TZIC registers</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 16</td>
<td>TZSC1IE: illegal access interrupt enable for GTZC1 TZSC registers</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td></td>
<td>Bits 15:12 Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 11</td>
<td>EXTIIE: illegal access interrupt enable for EXTI</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 10</td>
<td>RCCIE: illegal access interrupt enable for RCC</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 9</td>
<td>PWRIE: illegal access interrupt enable for PWR</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 8</td>
<td>TAMPIE: illegal access interrupt enable for TAMP</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
</tbody>
</table>
Bit 7 **RTCIE**: illegal access interrupt enable for RTC
- 0: interrupt disabled
- 1: interrupt enabled

Bit 6 **SBSIE**: illegal access interrupt enable for SBS
- 0: interrupt disabled
- 1: interrupt enabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **OTFDEC1IE**: illegal access interrupt enable for OTFDEC1
- 0: interrupt disabled
- 1: interrupt enabled

Bit 3 **FLASHIE**: illegal access interrupt enable for FLASH memory
- 0: interrupt disabled
- 1: interrupt enabled

Bit 2 **FLASH_REGIE**: illegal access interrupt enable for FLASH registers
- 0: interrupt disabled
- 1: interrupt enabled

Bit 1 **GPDMA2IE**: illegal access interrupt enable for GPDMA2
- 0: interrupt disabled
- 1: interrupt enabled

Bit 0 **GPDMA1IE**: illegal access interrupt enable for GPDMA1
- 0: interrupt disabled
- 1: interrupt enabled

### 5.7.5 GTZC1 TZIC status register 1 (GTZC1_TZIC_SR1)

Address offset: 0x010
Reset value: 0x0000 0000

Secure privileged access only.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM2F</td>
<td>DTSF</td>
<td>UART1F</td>
<td>UART9F</td>
<td>UART8F</td>
<td>UART7F</td>
<td>DAC1F</td>
<td>HDMI2F</td>
<td>USART11F</td>
<td>USART10F</td>
<td>USART6F</td>
<td>CRSF</td>
<td>I3C1F</td>
<td>I2C2F</td>
<td>I2C1F</td>
<td>UART5F</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>UART4F</td>
<td>USART3F</td>
<td>USART2F</td>
<td>SPI1F</td>
<td>SPI2F</td>
<td>IWDGF</td>
<td>WWDG</td>
<td>TIM14F</td>
<td>TIM13F</td>
<td>TIM12F</td>
<td>TIM11F</td>
<td>TIM10F</td>
<td>TIM9F</td>
<td>TIM8F</td>
<td>TIM7F</td>
<td>TIM6F</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **LPTIM2F**: illegal access flag for LPTIM2
- 0: no illegal access event
- 1: illegal access event

Bit 30 **DTSF**: illegal access flag for DTS
- 0: no illegal access event
- 1: illegal access event
Bit 29  **UART12F**: illegal access flag for UART12
   0: no illegal access event
   1: illegal access event

Bit 28  **UART9F**: illegal access flag for UART9
   0: no illegal access event
   1: illegal access event

Bit 27  **UART8F**: illegal access flag for UART8
   0: no illegal access event
   1: illegal access event

Bit 26  **UART7F**: illegal access flag for UART7
   0: no illegal access event
   1: illegal access event

Bit 25  **DAC1F**: illegal access flag for DAC1
   0: no illegal access event
   1: illegal access event

Bit 24  **HDMICECF**: illegal access flag for HDMICEC
   0: no illegal access event
   1: illegal access event

Bit 23  **USART11F**: illegal access flag for USART11
   0: no illegal access event
   1: illegal access event

Bit 22  **USART10F**: illegal access flag for USART10
   0: no illegal access event
   1: illegal access event

Bit 21  **USART6F**: illegal access flag for USART6
   0: no illegal access event
   1: illegal access event

Bit 20  **CRSF**: illegal access flag for CRS
   0: no illegal access event
   1: illegal access event

Bit 19  **I3C1F**: illegal access flag for I3C1
   0: no illegal access event
   1: illegal access event

Bit 18  **I2C2F**: illegal access flag for I2C2
   0: no illegal access event
   1: illegal access event

Bit 17  **I2C1F**: illegal access flag for I2C1
   0: no illegal access event
   1: illegal access event

Bit 16  **UART5F**: illegal access flag for UART5
   0: no illegal access event
   1: illegal access event

Bit 15  **UART4F**: illegal access flag for UART4
   0: no illegal access event
   1: illegal access event
Bit 14 **USART3F**: illegal access flag for USART3
0: no illegal access event
1: illegal access event

Bit 13 **USART2F**: illegal access flag for USART2
0: no illegal access event
1: illegal access event

Bit 12 **SPI3F**: illegal access flag for SPI3
0: no illegal access event
1: illegal access event

Bit 11 **SPI2F**: illegal access flag for SPI2
0: no illegal access event
1: illegal access event

Bit 10 **IWDGF**: illegal access flag for IWDG
0: no illegal access event
1: illegal access event

Bit 9 **WWDGF**: illegal access flag for WWDG
0: no illegal access event
1: illegal access event

Bit 8 **TIM14F**: illegal access flag for TIM14
0: no illegal access event
1: illegal access event

Bit 7 **TIM13F**: illegal access flag for TIM13
0: no illegal access event
1: illegal access event

Bit 6 **TIM12F**: illegal access flag for TIM12
0: no illegal access event
1: illegal access event

Bit 5 **TIM7F**: illegal access flag for TIM7
0: no illegal access event
1: illegal access event

Bit 4 **TIM6F**: illegal access flag for TIM6
0: no illegal access event
1: illegal access event

Bit 3 **TIM5F**: illegal access flag for TIM5
0: no illegal access event
1: illegal access event

Bit 2 **TIM4F**: illegal access flag for TIM4
0: no illegal access event
1: illegal access event

Bit 1 **TIM3F**: illegal access flag for TIM3
0: no illegal access event
1: illegal access event

Bit 0 **TIM2F**: illegal access flag for TIM2
0: no illegal access event
1: illegal access event
5.7.6 GTZC1 TZIC status register 2 (GTZC1_TZIC_SR2)

Address offset: 0x014
Reset value: 0x0000 0000

Secure privileged access only.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>LPTIM5F</th>
<th>Bit 30</th>
<th>LPTIM4F</th>
<th>Bit 29</th>
<th>LPTIM3F</th>
<th>Bit 28</th>
<th>LPTIM1F</th>
<th>Bit 27</th>
<th>I2C4F</th>
<th>Bit 26</th>
<th>I2C3F</th>
<th>Bit 25</th>
<th>LPUART1F</th>
<th>Bit 24</th>
<th>SPI5F</th>
<th>Bits 23:20</th>
<th>Bit 19</th>
<th>USBF</th>
<th>Bit 18</th>
<th>SAI2F</th>
<th>Bit 17</th>
<th>SAI1F</th>
<th>Bit 16</th>
<th>SPI6F</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>LPTIM5F</td>
<td>30</td>
<td>LPTIM4F</td>
<td>29</td>
<td>LPTIM3F</td>
<td>28</td>
<td>LPTIM1F</td>
<td>27</td>
<td>I2C4F</td>
<td>26</td>
<td>I2C3F</td>
<td>25</td>
<td>LPUART1F</td>
<td>24</td>
<td>SPI5F</td>
<td>23</td>
<td>22</td>
<td>Res.</td>
<td>21</td>
<td>Res.</td>
<td>20</td>
<td>Res.</td>
<td>19</td>
<td>USBF</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **LPTIM5F**: illegal access flag for LPTIM5
0: no illegal access event
1: illegal access event

Bit 30 **LPTIM4F**: illegal access flag for LPTIM4
0: no illegal access event
1: illegal access event

Bit 29 **LPTIM3F**: illegal access flag for LPTIM3
0: no illegal access event
1: illegal access event

Bit 28 **LPTIM1F**: illegal access flag for LPTIM1
0: no illegal access event
1: illegal access event

Bit 27 **I2C4F**: illegal access flag for I2C4
0: no illegal access event
1: illegal access event

Bit 26 **I2C3F**: illegal access flag for I2C3
0: no illegal access event
1: illegal access event

Bit 25 **LPUART1F**: illegal access flag for LPUART
0: no illegal access event
1: illegal access event

Bit 24 **SPI5F**: illegal access flag for SPI5
0: no illegal access event
1: illegal access event

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

Bit 19 **USBF**: illegal access flag for USB
0: no illegal access event
1: illegal access event

Bit 18 **SAI2F**: illegal access flag for SAI2
0: no illegal access event
1: illegal access event
Bit 17  **SAI1F**: illegal access flag for SAI1
        0: no illegal access event
        1: illegal access event

Bit 16  **SPI6F**: illegal access flag for SPI6
        0: no illegal access event
        1: illegal access event

Bit 15  **SPI4F**: illegal access flag for SPI4
        0: no illegal access event
        1: illegal access event

Bit 14  **TIM17F**: illegal access flag for TIM17
        0: no illegal access event
        1: illegal access event

Bit 13  **TIM16F**: illegal access flag for TIM16
        0: no illegal access event
        1: illegal access event

Bit 12  **TIM15F**: illegal access flag for TIM15
        0: no illegal access event
        1: illegal access event

Bit 11  **USART1F**: illegal access flag for USART1
        0: no illegal access event
        1: illegal access event

Bit 10  **TIM8F**: illegal access flag for TIM8
        0: no illegal access event
        1: illegal access event

Bit 9   **SPI1F**: illegal access flag for SPI1
        0: no illegal access event
        1: illegal access event

Bit 8   **TIM1F**: illegal access flag for TIM1
        0: no illegal access event
        1: illegal access event

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

Bit 2   **UCPDF**: illegal access flag for UCPD
        0: no illegal access event
        1: illegal access event

Bit 1   **FDCAN2F**: illegal access flag for FDCAN2
        0: no illegal access event
        1: illegal access event

Bit 0   **FDCAN1F**: illegal access flag for FDCAN1
        0: no illegal access event
        1: illegal access event
### 5.7.7 GTZC1 TZIC status register 3 (GTZC1_TZIC_SR3)

Address offset: 0x018  
Reset value: 0x0000 0000  
Secure privileged access only.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:27</td>
<td>Reserved, must be kept at reset value.</td>
<td>0000</td>
</tr>
<tr>
<td>26</td>
<td>RAMCFGF: illegal access flag for RAMSCFG</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>OCTOSPI1F: illegal access flag for OCTOSPI1</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>FMCF: illegal access flag for FMC</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>SDMMC2F: illegal access flag for SDMMC2</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>SDMMC1F: illegal access flag for SDMMC1</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>PKAF: illegal access flag for PKA</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>SAESF: illegal access flag for SAES</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>RNGF: illegal access flag for RNG</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>HASHF: illegal access flag for HASH</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>16:14</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>13:9</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>8:4</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>3:0</td>
<td></td>
<td>r</td>
</tr>
</tbody>
</table>

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

- Bit 26 **RAMCFGF**: illegal access flag for RAMSCFG  
  0: no illegal access event  
  1: illegal access event

- Bit 25 Reserved, must be kept at reset value.

- Bit 24 **OCTOSPI1F**: illegal access flag for OCTOSPI1  
  0: no illegal access event  
  1: illegal access event

- Bit 23 **FMCF**: illegal access flag for FMC  
  0: no illegal access event  
  1: illegal access event

- Bit 22 **SDMMC2F**: illegal access flag for SDMMC2  
  0: no illegal access event  
  1: illegal access event

- Bit 21 **SDMMC1F**: illegal access flag for SDMMC1  
  0: no illegal access event  
  1: illegal access event

- Bit 20 **PKAF**: illegal access flag for PKA  
  0: no illegal access event  
  1: illegal access event

- Bit 19 **SAESF**: illegal access flag for SAES  
  0: no illegal access event  
  1: illegal access event

- Bit 18 **RNGF**: illegal access flag for RNG  
  0: no illegal access event  
  1: illegal access event

- Bit 17 **HASHF**: illegal access flag for HASH  
  0: no illegal access event  
  1: illegal access event
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 16</td>
<td><strong>AESF</strong>: illegal access flag for AES</td>
<td>no illegal access event</td>
<td>illegal access event</td>
</tr>
<tr>
<td>Bit 15</td>
<td><strong>DCMIF</strong>: illegal access flag for DCMI</td>
<td>no illegal access event</td>
<td>illegal access event</td>
</tr>
<tr>
<td>Bit 14</td>
<td><strong>ADC12F</strong>: illegal access flag for ADC1 and ADC2</td>
<td>no illegal access event</td>
<td>illegal access event</td>
</tr>
<tr>
<td>Bit 13</td>
<td><strong>DCACHEF</strong>: illegal access flag for DCACHE</td>
<td>no illegal access event</td>
<td>illegal access event</td>
</tr>
<tr>
<td>Bit 12</td>
<td><strong>ICACHEF</strong>: illegal access flag for ICACHE</td>
<td>no illegal access event</td>
<td>illegal access event</td>
</tr>
<tr>
<td>Bit 11</td>
<td><strong>ETHF</strong>: illegal access flag for register of ETH</td>
<td>no illegal access event</td>
<td>illegal access event</td>
</tr>
<tr>
<td>Bit 10</td>
<td><strong>FMACF</strong>: illegal access flag for FMAC</td>
<td>no illegal access event</td>
<td>illegal access event</td>
</tr>
<tr>
<td>Bit 9</td>
<td><strong>CORDICF</strong>: illegal access flag for CORDIC</td>
<td>no illegal access event</td>
<td>illegal access event</td>
</tr>
<tr>
<td>Bit 8</td>
<td><strong>CRCF</strong>: illegal access flag for CRC</td>
<td>no illegal access event</td>
<td>illegal access event</td>
</tr>
<tr>
<td>Bits 7:2</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 1</td>
<td><strong>VREFBUFF</strong>: illegal access flag for VREFBUF</td>
<td>no illegal access event</td>
<td>illegal access event</td>
</tr>
<tr>
<td>Bit 0</td>
<td><strong>LPTIM6F</strong>: illegal access flag for LPTIM6</td>
<td>no illegal access event</td>
<td>illegal access event</td>
</tr>
</tbody>
</table>
### 5.7.8 GTZC1 TZIC status register 4 (GTZC1_TZIC_SR4)

Address offset: 0x01C
Reset value: 0x0000 0000
Secure privileged access only.

<table>
<thead>
<tr>
<th>Bit 31:30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23:21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>MPCB B3_RE</td>
<td>SRAM3 F</td>
<td>MPCB B2_RE</td>
<td>SRAM2 F</td>
<td>MPCB B1_RE</td>
<td>SRAM1 F</td>
<td>Reserved</td>
<td>BKP/AMF</td>
<td>FMC_MEMF</td>
<td>OCTOS_P11_MEMF</td>
<td>TZIC1F</td>
<td>TZSC1 F</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Reserved</td>
<td>EXTIF</td>
<td>RCCF</td>
<td>PWRF</td>
<td>TAMPF</td>
<td>RTCF</td>
<td>SBSF</td>
<td>Reserved</td>
<td>OTFDE C1F</td>
<td>FLASH F</td>
<td>FLASH _REGF</td>
<td>GPDM A2F</td>
<td>GPDM A1F</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:30 Reserved, must be kept at reset value.

- **Bit 29 MPCB B3_REGF:** illegal access flag for MPCB B3 registers
  - 0: no illegal access event
  - 1: illegal access event

- **Bit 28 SRAM3 F:** illegal access flag for SRAM3
  - 0: no illegal access event
  - 1: illegal access event

- **Bit 27 MPCB B2_REGF:** illegal access flag for MPCB B2 registers
  - 0: no illegal access event
  - 1: illegal access event

- **Bit 26 SRAM2 F:** illegal access flag for SRAM2
  - 0: no illegal access event
  - 1: illegal access event

- **Bit 25 MPCB B1_REGF:** illegal access flag for MPCB B1 registers
  - 0: no illegal access event
  - 1: illegal access event

- **Bit 24 SRAM1 F:** illegal access flag for SRAM1
  - 0: no illegal access event
  - 1: illegal access event

Bits 23:21 Reserved, must be kept at reset value.

- **Bit 20 BKP/AMF:** illegal access flag for MPCW4 (BKP/AMF) memory bank
  - 0: no illegal access event
  - 1: illegal access event

- **Bit 19 FMC_MEMF:** illegal access flag for MPCW2 (FMC_NOR bank), MPCW3 (FMC NAND bank and FMC SDRAM bank 1), and MPCW4 (FMC SDRAM bank 2)
  - 0: no illegal access event
  - 1: illegal access event

- **Bit 18 OCTOSP1_MEMF:** illegal access flag for MPCW1 (OCTOSP1) memory bank
  - 0: no illegal access event
  - 1: illegal access event
Bit 17 **TZIC1F**: illegal access flag for GTZC1 TZIC registers
0: no illegal access event
1: illegal access event

Bit 16 **TZSC1F**: illegal access flag for GTZC1 TZSC registers
0: no illegal access event
1: illegal access event

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

Bit 11 **EXTIF**: illegal access flag for EXTI
0: no illegal access event
1: illegal access event

Bit 10 **RCCF**: illegal access flag for RCC
0: no illegal access event
1: illegal access event

Bit 9 **PWRF**: illegal access flag for PWR
0: no illegal access event
1: illegal access event

Bit 8 **TAMPF**: illegal access flag for TAMP
0: no illegal access event
1: illegal access event

Bit 7 **RTCF**: illegal access flag for RTC
0: no illegal access event
1: illegal access event

Bit 6 **SBSF**: illegal access flag for SBS
0: no illegal access event
1: illegal access event

Bit 5 Reserved, must be kept at reset value.

Bit 4 **OTFDEC1F**: illegal access flag for OTFDEC1
0: no illegal access event
1: illegal access event

Bit 3 **FLASHF**: illegal access flag for FLASH memory
0: no illegal access event
1: illegal access event

Bit 2 **FLASH_REGF**: illegal access flag for FLASH registers
0: no illegal access event
1: illegal access event

Bit 1 **GPDMA2F**: illegal access flag for GPDMA2
0: no illegal access event
1: illegal access event

Bit 0 **GPDMA1F**: illegal access flag for GPDMA1
0: no illegal access event
1: illegal access event
### 5.7.9 GTZC1 TZIC flag clear register 1 (GTZC1_TZIC_FCR1)

Address offset: 0x020
Reset value: 0x0000 0000

Secure privileged access only.

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>CLPTIME2F</td>
<td>clear the illegal access flag for LPTIM2</td>
</tr>
<tr>
<td></td>
<td>0: no action</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: status flag cleared</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>CDTSF</td>
<td>clear the illegal access flag for DTS</td>
</tr>
<tr>
<td></td>
<td>0: no action</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: status flag cleared</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>CUART12F</td>
<td>clear the illegal access flag for UART12</td>
</tr>
<tr>
<td></td>
<td>0: no action</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: status flag cleared</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>CUART9F</td>
<td>clear the illegal access flag for UART9</td>
</tr>
<tr>
<td></td>
<td>0: no action</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: status flag cleared</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>CUART8F</td>
<td>clear the illegal access flag for UART8</td>
</tr>
<tr>
<td></td>
<td>0: no action</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: status flag cleared</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>CUART7F</td>
<td>clear the illegal access flag for UART7</td>
</tr>
<tr>
<td></td>
<td>0: no action</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: status flag cleared</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>CDAC1F</td>
<td>clear the illegal access flag for DAC1</td>
</tr>
<tr>
<td></td>
<td>0: no action</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: status flag cleared</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>CHDMICECF</td>
<td>clear the illegal access flag for HDMICEC</td>
</tr>
<tr>
<td></td>
<td>0: no action</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: status flag cleared</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>CUSART11F</td>
<td>clear the illegal access flag for USART11</td>
</tr>
<tr>
<td></td>
<td>0: no action</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: status flag cleared</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>CUSART10F</td>
<td>clear the illegal access flag for USART10</td>
</tr>
<tr>
<td></td>
<td>0: no action</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: status flag cleared</td>
<td></td>
</tr>
</tbody>
</table>

Secure privileged access only.
Bit 21 **CUSART6F**: clear the illegal access flag for USART6
   0: no action
   1: status flag cleared

Bit 20 **CCRSF**: clear the illegal access flag for CRS
   0: no action
   1: status flag cleared

Bit 19 **CI3C1F**: clear the illegal access flag for I3C1
   0: no action
   1: status flag cleared

Bit 18 **CI2C2F**: clear the illegal access flag for I2C2
   0: no action
   1: status flag cleared

Bit 17 **CI2C1F**: clear the illegal access flag for I2C1
   0: no action
   1: status flag cleared

Bit 16 **CUART5F**: clear the illegal access flag for UART5
   0: no action
   1: status flag cleared

Bit 15 **CUART4F**: clear the illegal access flag for UART4
   0: no action
   1: status flag cleared

Bit 14 **CUSART3F**: clear the illegal access flag for USART3
   0: no action
   1: status flag cleared

Bit 13 **CUSART2F**: clear the illegal access flag for USART2
   0: no action
   1: status flag cleared

Bit 12 **CSPI3F**: clear the illegal access flag for SPI3
   0: no action
   1: status flag cleared

Bit 11 **CSPI2F**: clear the illegal access flag for SPI2
   0: no action
   1: status flag cleared

Bit 10 **CIWDGF**: clear the illegal access flag for IWDG
   0: no action
   1: status flag cleared

Bit 9 **CWWDGF**: clear the illegal access flag for WWDG
   0: no action
   1: status flag cleared

Bit 8 **CTIM14F**: clear the illegal access flag for TIM14
   0: no action
   1: status flag cleared

Bit 7 **CTIM13F**: clear the illegal access flag for TIM13
   0: no action
   1: status flag cleared
Bit 6 **CTIM12F**: clear the illegal access flag for TIM12
0: no action
1: status flag cleared

Bit 5 **CTIM7F**: clear the illegal access flag for TIM7
0: no action
1: status flag cleared

Bit 4 **CTIM6F**: clear the illegal access flag for TIM6
0: no action
1: status flag cleared

Bit 3 **CTIM5F**: clear the illegal access flag for TIM5
0: no action
1: status flag cleared

Bit 2 **CTIM4F**: clear the illegal access flag for TIM4
0: no action
1: status flag cleared

Bit 1 **CTIM3F**: clear the illegal access flag for TIM3
0: no action
1: status flag cleared

Bit 0 **CTIM2F**: clear the illegal access flag for TIM2
0: no action
1: status flag cleared

### 5.7.10 GTZC1 TZIC flag clear register 2 (GTZC1_TZIC_FCR2)

Address offset: 0x024

Reset value: 0x0000 0000

Secure privileged access only.

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLPTIM5F</td>
<td>CLPTIM4F</td>
<td>CLPTIM3F</td>
<td>CLPTIM1F</td>
<td>CI2C4F</td>
<td>CI2C3F</td>
<td>CLPUA</td>
<td>RT1F</td>
<td>CSP15F</td>
<td>Res.</td>
<td>Res.</td>
<td>Res.</td>
<td>Res.</td>
<td>CUSBF</td>
<td>CSAI2F</td>
<td>CSAI1F</td>
<td>CSP16F</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 31 **CLPTIM5F**: clear the illegal access flag for LPTIM5
0: no action
1: status flag cleared

Bit 30 **CLPTIM4F**: clear the illegal access flag for LPTIM4
0: no action
1: status flag cleared

Bit 29 **CLPTIM3F**: clear the illegal access flag for LPTIM3
0: no action
1: status flag cleared
Bit 28 CLPTIM1F: clear the illegal access flag for LPTIM1
 0: no action
 1: status flag cleared

Bit 27 CI2C4F: clear the illegal access flag for I2C4
 0: no action
 1: status flag cleared

Bit 26 CI2C3F: clear the illegal access flag for I2C3
 0: no action
 1: status flag cleared

Bit 25 CLPUART1F: clear the illegal access flag for LPUART
 0: no action
 1: status flag cleared

Bit 24 CSPi5F: clear the illegal access flag for SPI5
 0: no action
 1: status flag cleared

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

Bit 19 CUSBF: clear the illegal access flag for USB
 0: no action
 1: status flag cleared

Bit 18 CSAI2F: clear the illegal access flag for SAI2
 0: no action
 1: status flag cleared

Bit 17 CSAI1F: clear the illegal access flag for SAI1
 0: no action
 1: status flag cleared

Bit 16 CSPi6F: clear the illegal access flag for SPI6
 0: no action
 1: status flag cleared

Bit 15 CSPi4F: clear the illegal access flag for SPI4
 0: no action
 1: status flag cleared

Bit 14 CTIM17F: clear the illegal access flag for TIM17
 0: no action
 1: status flag cleared

Bit 13 CTIM16F: clear the illegal access flag for TIM16
 0: no action
 1: status flag cleared

Bit 12 CTIM15F: clear the illegal access flag for TIM15
 0: no action
 1: status flag cleared

Bit 11 CUSART1F: clear the illegal access flag for USART1
 0: no action
 1: status flag cleared
Bit 10 **CTIM8F**: clear the illegal access flag for TIM8  
0: no action  
1: status flag cleared

Bit 9 **CSP1F**: clear the illegal access flag for SPI1  
0: no action  
1: status flag cleared

Bit 8 **CTIM1F**: clear the illegal access flag for TIM1  
0: no action  
1: status flag cleared

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

Bit 2 **CUCPDF**: clear the illegal access flag for UCPD  
0: no action  
1: status flag cleared

Bit 1 **CFDCAN2F**: clear the illegal access flag for FDCAN2  
0: no action  
1: status flag cleared

Bit 0 **CFDCAN1F**: clear the illegal access flag for FDCAN1  
0: no action  
1: status flag cleared

5.7.11 **GTZC1 TZIC flag clear register 3 (GTZC1_TZIC_FCR3)**

Address offset: 0x028  
Reset value: 0x0000 0000  
Secure privilege access only.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CDCMIF</td>
<td>CADC1 2F</td>
<td>CDCA</td>
<td>CHEF</td>
<td>CICAC</td>
<td>HEF</td>
<td>CETHF</td>
<td>CFMCF</td>
<td>CCORCF</td>
<td>CCRCF</td>
<td>CCOR</td>
<td>CCRC</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

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

Bit 26 **CRAMCFGF**: clear illegal access flag for RAMSCFG  
0: no action  
1: status flag cleared

Bit 25 Reserved, must be kept at reset value.

Bit 24 **COTCTOSPI1F**: clear illegal access flag for OCTOSPI1  
0: no action  
1: status flag cleared
Bit 23 **CFMCF**: clear illegal access flag for FMC
   0: no action
   1: status flag cleared

Bit 22 **CSDMMC2F**: clear illegal access flag for SDMMC2
   0: no action
   1: status flag cleared

Bit 21 **CSDMMC1F**: clear illegal access flag for SDMMC1
   0: no action
   1: status flag cleared

Bit 20 **CPKAF**: clear illegal access flag for PKA
   0: no action
   1: status flag cleared

Bit 19 **CSAESF**: clear illegal access flag for SAES
   0: no action
   1: status flag cleared

Bit 18 **CRNGF**: clear illegal access flag for RNG
   0: no action
   1: status flag cleared

Bit 17 **CHASHF**: clear illegal access flag for HASH
   0: no action
   1: status flag cleared

Bit 16 **CAESF**: clear illegal access flag for AES
   0: no action
   1: status flag cleared

Bit 15 **CDCMIF**: clear illegal access flag for DCMI
   0: no action
   1: status flag cleared

Bit 14 **CADC12F**: clear illegal access flag for ADC1 and ADC2
   0: no action
   1: status flag cleared

Bit 13 **CDCACHEF**: clear illegal access flag for DCACHE
   0: no action
   1: status flag cleared

Bit 12 **CICACHEF**: clear illegal access flag for ICACHE
   0: no action
   1: status flag cleared

Bit 11 **CETHF**: clear illegal access flag for register of ETH
   0: no action
   1: status flag cleared

Bit 10 **CFMACF**: clear illegal access flag for FMAC
   0: no action
   1: status flag cleared

Bit 9 **CCORDICF**: clear illegal access flag for CORDIC
   0: no action
   1: status flag cleared
Global TrustZone® controller (GTZC) RM0481

5.7.12 GTZC1 TZIC flag clear register 4 (GTZC1_TZIC_FCR4)

Address offset: 0x02C
Reset value: 0x0000 0000
Secure privilege access only.

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

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **CMPCBB3_REGF**: clear illegal access flag for MPCBB3 registers
0: no action
1: status flag cleared

Bit 28 **CSRAM3F**: clear illegal access flag for SRAM3
0: no action
1: status flag cleared

Bit 27 **CMPCBB2_REGF**: clear illegal access flag for MPCBB2 registers
0: no action
1: status flag cleared

Bit 26 **CSRAM2F**: clear illegal access flag for SRAM2
0: no action
1: status flag cleared

Bit 25 **CMPCBB1_REGF**: clear illegal access flag for MPCBB1 registers
0: no action
1: status flag cleared

Bit 24 **CSRAM1F**: clear illegal access flag for SRAM1
0: no action
1: status flag cleared
Bits 23:21  Reserved, must be kept at reset value.

Bit 20  **CBKPSRAMF**: clear the illegal access flag for MPCWM4 (BKPSRAM) memory bank
   0: no action
   1: status flag cleared

Bit 19  **CFMC_MEMF**: clear the illegal access flag for MPCWM2 (FMC_NOR bank), MPCWM3 (FMC_NAND bank and FMC_SDRAM bank 1), and MPCWM4 (FMC_SDRAM bank 2)
   0: no action
   1: status flag cleared

Bit 18  **C OCTOSPI1_MEMF**: clear the illegal access flag for MPCWM1 (OCTOSPI1) memory bank
   0: no action
   1: status flag cleared

Bit 17  **CTZIC1F**: clear the illegal access flag for GTZC1 TZIC registers
   0: no action
   1: status flag cleared

Bit 16  **CTZSC1F**: clear the illegal access flag for GTZC1 TZSC registers
   0: no action
   1: status flag cleared

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

Bit 11  **CEXTIF**: clear the illegal access flag for EXTI
   0: no action
   1: status flag cleared

Bit 10  **CRCFF**: clear the illegal access flag for RCC
   0: no action
   1: status flag cleared

Bit 9  **CPWRF**: clear the illegal access flag for PWR
   0: no action
   1: status flag cleared

Bit 8  **CTAMPF**: clear the illegal access flag for TAMP
   0: no action
   1: status flag cleared

Bit 7  **CRTCF**: clear the illegal access flag for RTC
   0: no action
   1: status flag cleared

Bit 6  **CSBSF**: clear the illegal access flag for SBS
   0: no action
   1: status flag cleared

Bit 5  Reserved, must be kept at reset value.

Bit 4  **COTFDEC1F**: clear the illegal access flag for OTFDEC1
   0: no action
   1: status flag cleared

Bit 3  **CFLASHF**: clear the illegal access flag for FLASH memory
   0: no action
   1: status flag cleared
Bit 2 **CFLASH_REGF**: clear the illegal access flag for FLASH registers
   0: no action
   1: status flag cleared

Bit 1 **CGPDM2A2F**: clear the illegal access flag for GPDMA2
   0: no action
   1: status flag cleared

Bit 0 **CGPDM2A1F**: clear the illegal access flag for GPDMA1
   0: no action
   1: status flag cleared
### Table 31. GTZC1 TZIC register map and reset values

| Offset  | Register                  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|---------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000   | GTZC1_TZIC_IER1           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|         | Reset value               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x004   | GTZC1_TZIC_IER2           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|         | Reset value               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x008   | GTZC1_TZIC_IER3           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|         | Reset value               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x00C   | GTZC1_TZIC_IER4           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|         | Reset value               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x010   | GTZC1_TZIC_SR1            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|         | Reset value               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x014   | GTZC1_TZIC_SR2            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|         | Reset value               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x018   | GTZC1_TZIC_SR3            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|         | Reset value               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x01C   | GTZC1_TZIC_SR4            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|         | Reset value               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x020   | GTZC1_TZIC_FCR1           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|         | Reset value               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| Offset | Register               | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x024  | GTZC1_TZIC_FCR2        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                       | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x028  | GTZC1_TZIC_FCR3        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                       | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x02C  | GTZC1_TZIC_FCR4        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                       | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Table 24: GTZC1 sub-block address offset.
5.8 GTZC1 MPCBBz registers (z = 1 to 3)

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

5.8.1 GTZC1 SRAMz MPCBB control register (GTZC1_MPCBBz_CR) (z = 1 to 3)

Address offset: 0x000
Reset value: 0x0000 0000
Secure privileged access only.

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

Bit 31 **SRWILADIS**: secure read/write illegal access disable
This bit disables the detection of an illegal access when a secure read/write transaction access a non-secure blocks of the block-based SRAM (secure fetch on non-secure block is always considered illegal).
0: enabled, secure read/write access not allowed on non-secure SRAM block
1: disabled, secure read/write access allowed on non-secure SRAM block

Bit 30 **INVSECSTATE**: SRAMx clocks security state
This bit is used to define the internal SRAMs clocks control in RCC as secure or not.
0: SRAMs clocks are secure if a secure area exists in the MPCBB. It is non secure if there is no secure area.
1: SRAMs clocks are non-secure even if a secure area exists in the MPCBB, and secure even if no secure block is set in the MPCBB.

Bits 29:1 Reserved, must be kept at reset value.

Bit 0 **GLOCK**: lock the control register of the MPCBB until next reset
This bit is cleared by default and once set, it can not be reset until system reset.
0: control register not locked
1: control register locked
5.8.2  GTZC1 SRAMz MPCBB configuration lock register 1
(GTZC1_MPCBBz_CFGLOCK1) (z = 1 to 3)

Address offset: 0x010
Reset value: 0x0000 0000
Secure privileged access only.

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

Bits 31:0  SPLCKy: Security/privilege configuration lock for super-block (y = 31 to 0)
This bit is set by software and can be cleared only by system reset.
0: GTZC1_MPCBBz_SECCFGRy and GTZC1_MPCBBz_PRIVCFGRy can be written.
1: Writes to GTZC1_MPCBBz_SECCFGRy and GTZC1_MPCBBz_PRIVCFGRy are ignored.

5.8.3  GTZC1 SRAMz MPCBB security configuration for super-block x
register (GTZC1_MPCBBz_SECCFGRx) (z = 1 to 3)

Address offset: 0x100 + 0x4 * x, (x = 0 to 31)
Reset value: 0xFFFF FFFF
The given reset value is valid when TZEN = 0xB4. The reset value is 0x0000 0000 when TZEN = 0xC3.
Write access to this register is secure only. Any read is allowed.

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

Bits 31:0  SECy: Security configuration for block y (y = 31 to 0)
0: Non-secure access only to block y, belonging to super-block x. Secure access is also allowed if the SRWILADIS bit is set in GTZC1_MPCBBz_CR.
1: Secure access only to block y, belonging to super-block x.
Unprivileged write to this bit is ignored if PRIVy bit is set in GTZC1_MPCBBz_PRIVCFGRx.
Writes are ignored if SPLCKx bit is set in GTZC1_MPCBBz_CFGLOCK.
5.8.4  **GTZC1 SRAMz MPCBB privileged configuration for super-block x register (GTZC1_MPCBBz_PRIVCFGRx) (z = 1 to 3)**

Address offset: 0x200 + 0x * x, (x = 0 to 31)

Reset value: 0xFFFF FFFF

The given reset value is valid when TZEN = 0xB4. The reset value is 0x0000 0000 when TZEN = 0xC3. Write access to this register is privileged only. Any read is allowed.

<table>
<thead>
<tr>
<th>Address offset: 0x200 + 0x * x, (x = 0 to 31)</th>
<th>Reset value</th>
<th>Bits 31:0 PRIVy: Privileged configuration for block y, belonging to super-block x (y = 31 to 0).</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000 GTZC1_MPCBBz_CR</td>
<td>0</td>
<td>0: Privileged and unprivileged access to block y, belonging to super-block x</td>
</tr>
<tr>
<td>Reserved</td>
<td>0x010 GTZC1_MPCBBz_CFGLOCK1</td>
<td>1: Only privileged access to block y, belonging to super-block x</td>
</tr>
<tr>
<td>Reserved</td>
<td>0x100 + 0x04 *x (x = 0 to 31) GTZC1_MPCBBz_SECCFGRx</td>
<td>Non-secure write to this bit is ignored if SECy bit is set in GTZC1_MPCBBz_SECCFGRx. Writes are ignored if SPLCKx bit is set in GTZC1_MPCBBz_CFGLOCK.</td>
</tr>
</tbody>
</table>

5.8.5  **GTZC1 MPCBBz register map (z = 1 to 3)**

Refer to Table 24: GTZC1 sub-block address offset.
6 RAMs configuration controller (RAMCFG)

6.1 Introduction

The RAMCFG configures the features of the internal SRAMs (SRAM1, SRAM2, SRAM3, and BKPSRAM).

6.2 RAMCFG main features

The internal SRAM supports some of the features listed hereafter, configured in RAMCFG:

- Error code correction (ECC):
  - Single error detection and correction with interrupt generation
  - Double error detection with interrupt or NMI generation
  - Status with failing address
- Write protection (1-Kbyte granularity)
- SRAM software erase

6.3 RAMCFG functional description

6.3.1 Internal SRAMs features

Five SRAMs are embedded in the devices, each with specific features:

- SRAM1, SRAM2, SRAM3 are the main SRAMs.
  These SRAMs are made of several blocks that can be powered down in Stop mode to reduce consumption:
  - SRAM1: four 64-Kbyte blocks (total 256 Kbytes)
  - SRAM2: 16-Kbyte + 48-Kbyte blocks (total 64 Kbytes)
  - SRAM3: five 64-Kbyte blocks (total 320 Kbytes)

The backup SRAM (BKPSRAM) can be retained in all low-power modes and when \( V_{DD} \) is off in VBAT mode. Refer to Section 10: Power control (PWR) for more details.

- SRAM2 is erased when a system reset occurs if the SRAM2_RST option bit is selected in the Flash memory user option bytes. SRAM1 and SRAM3 are erased when a system reset occurs if the SRAM13_RST option bit is selected in the Flash memory user option bytes. Refer to Section 7: Embedded flash memory (FLASH) for more details.

- SRAM2 and optionally backup SRAM are protected by the tamper detection circuit, and are erased by hardware in case of tamper detection. Refer to Section 76: Tamper and backup registers (TAMP) (TBD) for more details.

- The RAMCFG embeds the registers related to the internal SRAMs ECC, write protection and software erase.
The table below summarizes the features supported by each internal SRAM

<table>
<thead>
<tr>
<th>SRAM feature</th>
<th>SRAM1 (256 Kbytes)</th>
<th>SRAM2 (64 Kbytes)</th>
<th>SRAM3 (320 Kbytes)</th>
<th>BKPSRAM (4 Kbytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional retention in Standby mode</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Optional retention in VBAT mode</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Erased with tamper detection and Backup domain reset</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>X(1)</td>
</tr>
<tr>
<td>Optionally erased with system reset</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Software erase</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ECC</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Write protection</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1. Optional: BKPSRAM can be configured to be erased or not on tamper detection.

### 6.3.2 Error code correction (SRAM2, SRAM3, BKPSRAM)

The ECC is supported by SRAM2, SRAM3 and BKPSRAM when enabled with the SRAM2_ECC, SRAM3_ECC and BKPRAM_ECC user option bits. Refer to Section 7: Embedded flash memory (FLASH) for more details.

Seven ECC bits are added per 32 bits of SRAM, allowing two bits error detection and one bit error correction on memory read access.

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

**Caution:** In case of a byte or half-word write on SRAM with ECC, the read/modify/write operation is done in a buffer. The buffer content is written into the SRAM two AHB clock cycles after the SRAM AHB is released (when SRAM is no more accessed).

### Single and double ECC errors

When a single error is detected, it is automatically corrected and the SEDC/CSEDCC bits are set in the RAMCFG memory x interrupt status register (RAMCFG_MxISR) and RAMCFG memory x interrupt clear register x (RAMCFG_MxICR) respectively. An interrupt is generated if enabled by the SEIE bit in the RAMCFG memory x interrupt enable register (RAMCFG_MxIER). The failing address is stored in the RAMCFG memory x ECC single error address register (RAMCFG_MxSEAR) if the ALE bit is set in the RAMCFG memory x control register (RAMCFG_MxCR).

**Caution:** Single errors cannot be detected when the SEDC bit is set.

When a double error is detected, the DED and CDED bits are set in the RAMCFG memory interrupt status register (RAMCFG_MxISR) and RAMCFG memory x interrupt clear register x (RAMCFG_MxICR) respectively. An interrupt or NMI is generated if enabled by the DEIE or ECCNMI bit in the RAMCFG memory x interrupt enable register (RAMCFG_MxIER). The failing address is stored in the RAMCFG memory x ECC double error address register.
**RAMs configuration controller (RAMCFG)**

*Note:* If the ALE bit is set in the **RAMCFG memory x control register (RAMCFG_MxCR)**.

**Caution:** Double errors cannot be detected when the DED bit is set.

### SRAM3 ECC specific management

When the ECC is enabled for SRAM3, only the first 256 Kbytes of SRAM3 are with ECC. The next 64 Kbytes are without ECC, and the last block is used to store the ECC, so cannot be used for application.

The figure below shows the SRAM areas, when SRAM2 and SRAM3 ECC are enabled.

#### Figure 19. SRAM1, SRAM2 with ECC and SRAM3 with ECC memory map

<table>
<thead>
<tr>
<th>Address offset</th>
<th>SRAM1</th>
<th>SRAM2</th>
<th>SRAM3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000-0x0400</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0x0400-0x04FF</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0x0500-0x0600</td>
<td>48 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0x0600-0x0700</td>
<td>16 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0x0700-0x09FF</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0x0A00-0x1FFF</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0x2000-0x3FFF</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0x4000-0x4FFF</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0x5000-0x5FFF</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0x6000-0x6FFF</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0x7000-0x7FFF</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0x8000-0x8FFF</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0x9000-0x9FFF</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0xA000-0xBFFF</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0xC000-0xCFFF</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0xD000-0xEFFF</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
<tr>
<td>0xF000-0xFFFF</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
<td>64 Kbytes</td>
</tr>
</tbody>
</table>

When ECC is enabled by user option bits, the ECCE bit is automatically set after system reset in the related **RAMCFG memory x control register (RAMCFG_MxCR)**.

The ECC can be deactivated by executing the following software sequence:

1. Write 0xAE in the **RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR)**.
2. Write 0x75 in the **RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR)**.
3. Write 0 in the ECCE bit of the **RAMCFG memory x control register (RAMCFG_MxCR)**.

When ECC is deactivated (ECCE = 0), the SRAM3 ECC storage area can be read and written for ECC user test purpose. When the ECC is activated (ECCE = 1), this area is reserved for ECC storage purpose and cannot be read nor written.
6.3.3 Write protection (SRAM2)

The SRAM2 is made of 64 1-Kbyte pages. Each 1-Kbyte page can be write-protected by setting its corresponding PxWP (x = 0 to 63) bit in the RAMCFG memory 2 write protection register 1 (RAMCFG_M2WPR1) and RAMCFG memory 2 write protection register 2 (RAMCFG_M2WPR2).

6.3.4 Software erase

SRAM erase can be requested by executing this software sequence:
1. Write 0xCA in the RAMCFG memory x erase key register (RAMCFG_MxERKEYR).
2. Write 0x53 in the RAMCFG memory x erase key register (RAMCFG_MxERKEYR).
3. Write 1 in the SRAMER bit of the RAMCFG memory x control register (RAMCFG_MxCR).

SRAMBUSY flag is set in the related SRAM interrupt status register as long as the erase is on going.

The total duration of each SRAM erase is N AHB clock cycles, where N is the size of the SRAM in 32-bit words.

If the SRAM is read or written while an erase is on going, wait states are inserted on the AHB bus until the end of the erase operation.

6.4 RAMCFG low-power modes

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

6.5 RAMCFG interrupts

The table below gives the list of RAMCFG interrupt requests.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit the Sleep mode</th>
<th>Exit the Stop mode</th>
<th>Exit the Standby modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAMCFG</td>
<td>ECC single error detection and correction</td>
<td>SEDC</td>
<td>SEIE</td>
<td>Write 1 in CSEDC</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>ECC double error detection</td>
<td>DED</td>
<td>DEIE = 1 and ECCNMI = 0</td>
<td>Write 1 in CDED</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NMI</td>
<td>ECC double error detection</td>
<td>DED</td>
<td>ECCNMI</td>
<td>Write 1 in CDED</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
6.6 **RAMCFG registers**

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

### 6.6.1 RAMCFG memory x control register (RAMCFG_MxCR)

Address offset: 0x040 * (x - 1), (x = 1, 2, 3, and 5)
Reset value: 0x0000 000X

ECCE reset value depends on ECC enable user option bit.

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

Bits 31:9 Reserved, must be kept at reset value.

**Bit 8** **SRAMER**: SRAM erase

This bit can be set by software only after writing the unlock sequence in the ERASEKEY field of the RAMCFG_MxERKEYR register. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.

- 0: No erase operation on going
- 1: Erase operation on going

Bits 7:5 Reserved, must be kept at reset value.

**Bit 4** **ALE**: Address latch enable

- 0: Failing address not stored in the SRAMx ECC single/double error address registers
- 1: Failing address stored in the SRAMx ECC single/double error address registers

*Note: This bit is reserved and must be kept at reset value in SRAM1 control register.*

Bits 3:1 Reserved, must be kept at reset value.

**Bit 0** **ECCE**: ECC enable.

This bit reset value is defined by the user option bit configuration. When set, it can be cleared by software only after writing the unlock sequence in the RAMCFG_MxECCKEYR register.

- 0: ECC disabled
- 1: ECC enabled

*Note: This bit is reserved and must be kept at reset value in SRAM1 control register.*
6.6.2 RAMCFG memory x interrupt enable register (RAMCFG_MxIER)

Address offset: 0x004 + 0x040 * (x - 1), (x = 2, 3, 5)
Reset value: 0x0000 0000

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

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

Bit 3 **ECCNMI**: Double error NMI
This bit is set by software and cleared only by a global RAMCFG reset.
0: NMI not generated in case of ECC double error
1: NMI generated in case of ECC double error

*Note*: if **ECCNMI** is set, the RAMCFG maskable interrupt is not generated whatever **DEIE** bit value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DEIE**: ECC double error interrupt enable
0: Double error interrupt disabled
1: Double error interrupt enabled

Bit 0 **SEIE**: ECC single error interrupt enable
0: Single error interrupt disabled
1: Single error interrupt enabled

6.6.3 RAMCFG memory interrupt status register (RAMCFG_MxISR)

Address offset: 0x008 + 0x040 * (x - 1), (x = 1, 2, 3 and 5)
Reset value: 0x0000 0000

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSEY**: SRAM busy with erase operation
0: No erase operation on going
1: Erase operation on going

*Note*: Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option bit is enabled, tamper detection or readout protection regression. Refer to **Table 33: Internal SRAMs features**.
Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **DED**: ECC double error detected
- 0: No double error
- 1: Double error detected

*Note: This bit is reserved and must be kept at reset value in SRAM1 interrupt status register.*

Bit 0 **SEDC**: ECC single error detected and corrected
- 0: No single error
- 1: Single error detected and corrected

*Note: This bit is reserved and must be kept at reset value in SRAM1 interrupt status register.*

### 6.6.4 RAMCFG memory x ECC single error address register (RAMCFG_MxSEAR)

Address offset: 0x00C + 0x040 \*(x - 1), (x = 2, 3, 5)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESEA[31:16]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>ESEA[15:0]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 **ESEA[31:0]**: ECC single error address

When the ALE bit is set in the RAMCFG_MxCR register, this field is updated with the address corresponding to the ECC single error.

### 6.6.5 RAMCFG memory x ECC double error address register (RAMCFG_MxDEAR)

Address offset: 0x010 + 0x040 \*(x - 1), (x = 2, 3, 5)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDEA[31:16]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>EDEA[15:0]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 **EDEA[31:0]**: ECC double error address

When the ALE bit is set in the RAMCFG_MxCR register, this field is updated with the address corresponding to the ECC double error.
### 6.6.6 RAMCFG memory x interrupt clear register x (RAMCFG_MxICR)

Address offset: $0x014 + 0x040 \times (x - 1)$, $(x = 2, 3, 5)$

Reset value: $0x0000\ 0000$

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

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

- **Bit 1 CDED**: Clear ECC double error detected
  - Writing 1 to this flag clears the DED bit in the RAMCFG_MxISR register. Reading this flag returns the DED value.

- **Bit 0 CSEDC**: Clear ECC single error detected and corrected
  - Writing 1 to this flag clears the SEDC bit in the RAMCFG_MxISR register. Reading this flag returns the SEDC value.

### 6.6.7 RAMCFG memory 2 write protection register 1 (RAMCFG_M2WPR1)

Address offset: $0x058$

Reset value: $0x0000\ 0000$

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>P31WP</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>P30WP</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>P29WP</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>P28WP</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>P27WP</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>P26WP</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>P25WP</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>P24WP</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>P23WP</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>P22WP</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>P21WP</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>P20WP</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>P19WP</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>P18WP</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>P17WP</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>P16WP</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>P15WP</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>P14WP</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>P13WP</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>P12WP</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>P11WP</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>P10WP</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>P9WP</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>P8WP</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>P7WP</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>P6WP</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>P5WP</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>P4WP</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>P3WP</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>P2WP</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>P1WP</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>P0WP</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **PyWP**: SRAM2 1-Kbyte page y write protection $(y = 31 \text{ to } 0)$

- These bits are set by software and cleared only by a global RAMCFG reset.
- 0: Write protection of SRAM2 1-Kbyte page y is disabled.
- 1: Write protection of SRAM2 1-Kbyte page y is enabled.
6.6.8 RAMCFG memory 2 write protection register 2 (RAMCFG_M2WPR2)

Address offset: 0x05C
Reset value: 0x0000 0000

```
          P60WP P59WP P58WP P57WP P56WP P55WP P54WP P53WP P52WP P51WP P50WP P49WP P48WP
  31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
rs rs rs rs rs rs rs rs rs rs rs rs rs rs
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
          P47WP P46WP P45WP P44WP P43WP P42WP P41WP P40WP P39WP P38WP P37WP P36WP P35WP P34WP P33WP P32WP
rs rs rs rs rs rs rs rs rs rs rs rs rs rs rs
```

Bits 31:0 PyWP: SRAM2 1-Kbyte page y write protection (y = 63 to 32)
These bits are set by software and cleared only by a global RAMCFG reset.
0: Write protection of SRAM2 1-Kbyte page y is disabled.
1: Write protection of SRAM2 1-Kbyte page y is enabled.

6.6.9 RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR)

Address offset: 0x024 + 0x040 * (x - 1), (x = 2, 3, 5)
Reset value: 0x0000 0000

```
          P40WP P39WP P38WP P37WP P36WP P35WP P34WP P33WP P32WP
  31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
rs rs rs rs rs rs rs rs rs rs rs rs rs rs rs
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
          ECCKEY[7:0]

  w w w w w w w w
```

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

Bits 7:0 ECCKEY[7:0]: ECC write protection key
The following steps are required to unlock the write protection of the ECCE bit in the RAMCFG_MxCR register.
1) Write 0xAE into ECCKEY[7:0].
2) Write 0x75 into ECCKEY[7:0].

Note: Writing a wrong key reactivates the write protection.
6.6.10 RAMCFG memory x erase key register (RAMCFG_MxERKEYR)

Address offset: 0x028 + 0x040 \(x - 1\), \(x = 1\) to 5
Reset value: 0x0000 0000

6.6.11 RAMCFG register map

Table 36. RAMCFG register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>RAMCFG_M1CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>RAMCFG_M1ISR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C to</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x28</td>
<td>RAMCFG_M1ERKEYR</td>
<td>ERASEKEY[7:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2C to</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x40</td>
<td>RAMCFG_M2CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x44</td>
<td>RAMCFG_M2IER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 36. RAMCFG register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>SRAMBUSY</th>
<th>ALE</th>
<th>ECCAM</th>
<th>DBE</th>
<th>SBE</th>
<th>ECCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x048</td>
<td>RAMCFG_M2ISR</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x04C</td>
<td>RAMCFG_M2SEAR</td>
<td>Reserved</td>
<td>Reserved</td>
<td>ESEA[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x050</td>
<td>RAMCFG_M2DEAR</td>
<td>Reserved</td>
<td>EDEA[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x054</td>
<td>RAMCFG_M2ICR</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x058</td>
<td>RAMCFG_M2WPR1</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x05C</td>
<td>RAMCFG_M2WPR2</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x060</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x064</td>
<td>RAMCFG_M2ECCKEYR</td>
<td>Reserved</td>
<td>ECCKEY[7:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x068</td>
<td>RAMCFG_M2ERKEYR</td>
<td>Reserved</td>
<td>ERASEKEY[7:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x06C to 0x067C</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x080</td>
<td>RAMCFG_M3CR</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x084</td>
<td>RAMCFG_M3IER</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x088</td>
<td>RAMCFG_M3ISR</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x08C</td>
<td>RAMCFG_M3SEAR</td>
<td>Reserved</td>
<td>Reserved</td>
<td>ESEA[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x090</td>
<td>RAMCFG_M3DEAR</td>
<td>Reserved</td>
<td>EDEA[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x094</td>
<td>RAMCFG_M3ICR</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x098 to 0x0A0</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
Table 36. RAMCFG register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>offset</th>
<th>reset value</th>
<th>ESEA[31:0]</th>
<th>EDEA[31:0]</th>
<th>SRAMBUSY</th>
<th>SPAMER</th>
<th>ECCE</th>
<th>DEIE</th>
<th>SEIE</th>
<th>DED</th>
<th>SEDC</th>
<th>CDED</th>
<th>CSDEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0A4</td>
<td>RAMCFG_M3ECCKEYR</td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0A8</td>
<td>RAMCFG_M3ERKEYR</td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0AC to 0x0FC</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x100</td>
<td>RAMCFG_M5CR</td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x104</td>
<td>RAMCFG_M5IER</td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x108</td>
<td>RAMCFG_M5ISR</td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10C</td>
<td>RAMCFG_M5SEAR</td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x110</td>
<td>RAMCFG_M5DEAR</td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x114</td>
<td>RAMCFG_M5ICR</td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x118 to 0x120</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x124</td>
<td>RAMCFG_M5ECCKEYR</td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x128</td>
<td>RAMCFG_M5ERKEYR</td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for the register boundary addresses.
7 Embedded flash memory (FLASH)

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

FLASH manages the automatic loading of nonvolatile user option bytes at power-on reset, and implements the dynamic update of these options. It also features a high-cycle data area, a one-time-programmable (OTP) area, a secure key storage area (OBKeys), and a read-only area configured by STMicroelectronics during manufacturing.

7.2 FLASH main features
- Up to 2 Mbytes of nonvolatile memory, divided into two 1 Mbyte banks
- Flash memory read operations supporting multiple lengths: 128 bits, 64 bits, 32 bits, 16 bits, or one byte
- Flash memory programming by 128 (user area, OBKeys) and by 16 bits (OTP and flash high-cycle data area)
- 8-Kbyte sector erase, bank erase and dual-bank mass erase
- Dual-bank organization supporting:
  - Simultaneous operations: read-while-write (program and erase) is supported, including flash high-cycle data area. The two banks share the same interface, hence write and erase cannot be performed in parallel (write-while-write is not supported).
  - Bank swapping: the address mapping of the user memory of each bank can be swapped, along with the corresponding registers. Security flags remain valid for the physical bank, so the data are not revealed by swapping to a bank with lower security configuration.
- Error code correction (ECC): one error detection/correction, or two errors detection per 128-bit flash word using nine ECC bits, on 16-bit words with six bits within configurable flash high-cycle data area.
- User configurable nonvolatile option bytes
- Flash memory enhanced protections, activated by option bytes
  - different product states for protecting memory content from debug access
  - sector group write-protection (WRPSG), protecting up to 32 groups of four sectors (32 Kbytes) per bank
  - two secure-only areas (one per user flash bank): when enabled, these areas are accessible only if the microcontroller operates in Secure access mode
  - HDP protection, providing temporal isolation for startup code
- 2-Kbyte one-time programmable (OTP) area
- Read-only area configured by STMicroelectronics
- Prefetch reads the next sequential instruction from flash memory
- Up to 48 Kbytes per bank supporting high-cycling capability (100 kcycles), to be used for data (EEPROM emulation)

7.3 FLASH functional description

7.3.1 FLASH block diagram

Figure 20 shows the embedded flash memory block diagram.
7.3.2 FLASH signals

The flash memory has two AHB connections, namely the flash AHB register interface and the main AHB interface.

Flash AHB register interface

- Data size is 32 bits
- Except for some registers (FLASH_NS/SECKEYR, FLASH_NS/SECOBKEYR and FLASH_OPTKEYR, used to insert unlock sequences for control, and option registers that can be written by 32 bits), it is possible to read and write all registers by 8, 16 and 32 bits.
- When unlock sequence for control and option registers is wrong, a bus error is raised, otherwise no read or write errors are generated on the bus.

Main AHB interface

The AHB data bus size is 128 bits. This interface is used to handle three different targets:

- Code placed in user and system memory. It is protected by 9 bits of ECC.
- Secure keys placed in OBKey sectors, and protected by 9 bits of ECC.
- OTP, read-only and flash memory high-cycle data area protected by 6 bits of ECC.

The main AHB interface is implemented as follows:

- User and system memory, OBKeys storage:
  - Supports multiple length: 128-, 64-, 32-, 16- and 8-bit data width.
  - There is a read buffer of 128 bits for each bank where the last data read is stored. If data are available in the read buffer, no read access is given to the flash memory. Buffer is flushed when write access, OTP access, OBK swap, OBK alternate sector erase, high-cycle data area access, user option change request or erase operation occur.
  - There is a prefetch of the same size as the read buffer.
  - A 9-bit ECC is associated to each 128-bit data flash memory word.

- OTP, read-only and flash high-cycle data
  - Two dedicated data buffer of 137 bits are used to manage 16-bit data with 6-bit ECC
  - Reading two times the same address triggers two flash read accesses.
  - During a read access, two wait states are added in addition to the memory wait states. These wait states are necessary to parse the data buffer.
  - Each write access triggers a write in the flash memory.
  - 6-bit ECC is associated to each 16-bit data.

By default, all the AHB memory range is cacheable. For regions where caching is not practical (OTP, RO, data area), MPU must be used to disable local cacheability.
7.3.3 Flash memory architecture and usage

Flash memory architecture

Figure 21 shows the organization supported by the embedded flash memory.

The embedded flash nonvolatile memory is composed of:

- A 2-Mbyte main memory block, organized in two banks, 1 Mbyte each. Each bank is divided in 128 sectors of 8 Kbytes each, and features flash-word rows of 128 bits + 9 bits of ECC per word.
- A system memory block of 128 Kbytes, divided into two 64-Kbyte banks. Each bank is divided in eight 8-Kbyte sectors. The system flash memory is ECC protected (9-bit ECC per 128-bit word).
- A set of nonvolatile option bytes loaded at reset by the embedded flash memory and accessible by the application software only through the AHB configuration register interface.
- A 2-Kbyte one-time-programmable (OTP) area that can be written only once by the application software.
- A 2-Kbyte read-only area. It contains a unique device ID and product information.
- Two memory sectors (2 x 8 Kbytes) of secure key storage, OBKeys.
- Up to 16 sectors of user flash memory with high cycling capability (100 K cycles) for data, 8 sectors per bank.

The overall flash memory architecture and its corresponding access interface is summarized in Table 40.
### Partition usage

*Figure 22* shows how the embedded flash memory is used by STMicroelectronics and by the application software.

#### Figure 22. Embedded flash memory usage

User and system memories are used differently, according to product state and other option bytes settings:

- **The user memory** contains the application code and data, while the system memory is used with root secure services (RSS), the debug authentication code, and the STM32 bootloader. When a reset occurs, the core jumps to the boot address configured through the BOOT pin, the (SEC/NS)BOOTADD option bytes and the product state.

- **The unique boot entry** (BOOT_UBE) is set to either ST iRoT located in the system flash memory, or iRoT in the user flash memory. This feature is available only on cryptography enabled devices.

- **Secure service library** in system flash memory is available in secure mode.

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

- **If (SEC/NS)BOOTADD is not yet configured**, dedicated libraries can be used for secure boot. They are located in the system flash memory:
  - ST libraries in system flash memory assist the application software boot with special features, such as secure boot and secure firmware install (SFI-RSS)
  - ST iROT (immutable root of trust) secure software in user flash memory is used for secure firmware update and provisioning (SFU)
Note: For more information on option byte setup for boot, refer to Section 7.4.6.

7.3.4 FLASH read operations

Read operation overview

Read access to main and system flash memory operates as follows:

- There is a 128 bit read data buffer associated to each bank, which stores the last data read. If several consecutive read accesses request data belonging to the same flash data word (128 bits), data are read directly from the current data read buffer, without triggering additional flash read operations. This mechanism occurs each time a read access is granted. When a read access is rejected for security reasons, the corresponding read error response is issued by the embedded flash memory and no read operation to flash memory is triggered.

- The read data buffer is disabled when write access or OTP access, user option change request, OBK swap, OBK erase or other erase operation occur.

Read access to OTP, RO and flash high-cycle data operates as follows:

1. Flash data word of 137 bits is read and stored in a temporary buffer.
2. The interface parses the 137 bits data word, and selects the 16- or 32-bit data requested.
3. While parsing the 137-bit data word, (two) wait states are added, and the AHB bus is stalled.
4. If the application reads an OTP data or flash high-cycle data not previously written, a double ECC error is reported and only a word full of set bits is returned (see Section 7.3.9 for details). The read data (in 16 bits) is stored in FLASH_ECCDR register, so that the user can identify if the double ECC error is due to a virgin data or a real ECC error.
5. Reading two times the same address triggers two reads in the flash memory.
6. For 8-bit accesses, an AHB bus error is generated.

Note: The embedded flash memory can perform single-error correction and double-error detection while read operations are being executed (see Section 7.3.8).

Instruction prefetch

The Cortex-M33 fetches instructions and literal pools (constants/data) over the C-Bus and through the instruction cache, if it is enabled. The prefetch block aims at increasing the efficiency of C-Bus accesses when the instruction cache is enabled, by reducing the cache refill latency.

Prefetch is efficient in case of sequential code; prefetch in the flash memory allows the next sequential instruction line to be read from the memory, while the current instruction line is being filled in instruction cache and executed by the CPU.

Prefetch is enabled by setting the PRFTEN bit in the FLASH access control register (FLASH_ACR). PRFTEN must be set only if at least one wait state is needed to access the flash memory.

Adjusting read timing constraints

The embedded clock must be enabled and running before reading data from a nonvolatile memory.
To correctly read data from the memory, the number of wait states (LATENCY) must be correctly programmed in the access control register (FLASH_ACR), according to the main AHB interface clock frequency, and the internal voltage range of the device ($V_{\text{core}}$).

Table 37 shows the correspondence between the number of wait states (LATENCY), the programming delay parameter (WRHIGHFREQ), the embedded flash memory clock frequency, and the supply voltage range.

Table 37. FLASH recommended number of wait states and programming delay

<table>
<thead>
<tr>
<th>Number of wait states (LATENCY)</th>
<th>Programming delay (WRHIGHFREQ)</th>
<th>Interface clock frequency vs. $V_{\text{CORE}}$ range$^{(1)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 WS (1 FLASH clock cycle)</td>
<td>00</td>
<td>VOS3 range 0.95 to 1.05 V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VOS2 range 1.05 to 1.15 V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VOS1 range 1.15 to 1.26 V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VOS0 range 1.30 to 1.40 V</td>
</tr>
<tr>
<td>1 WS (2 FLASH clock cycles)</td>
<td>00</td>
<td>0 to 20 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 to 30 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 to 34 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 to 42 MHz</td>
</tr>
<tr>
<td>2 WS (3 FLASH clock cycles)</td>
<td>01</td>
<td>20 to 40 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30 to 60 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>34 to 68 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>42 to 84 MHz</td>
</tr>
<tr>
<td>3 WS (4 FLASH clock cycles)</td>
<td>01</td>
<td>40 to 60 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>60 to 90 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>68 to 102 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>84 to 126 MHz</td>
</tr>
<tr>
<td>4 WS (5 FLASH clock cycles)</td>
<td>01</td>
<td>60 to 80 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>90 to 120 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>102 to 136 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>126 to 168 MHz</td>
</tr>
<tr>
<td>5 WS (6 FLASH clock cycles)</td>
<td>10</td>
<td>80 to 100 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>120 to 150 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>136 to 170 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>168 to 210 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>170 to 200 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>210 to 250 MHz</td>
</tr>
</tbody>
</table>

1. Voltage range from 1.26 to 1.30 V is not supported.

Adjusting system frequency

After power-on, the embedded flash memory is clocked by the 64 MHz high-speed internal oscillator (HSI), with a voltage range set at a scaled value of VOS3: a conservative 3 wait-state latency is specified in FLASH_ACR register (see Table 37).

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

To increase the CPU frequency:

1. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in the FLASH_ACR register, as described in Table 37.
2. Check that the new number of wait states is taken into account by reading back the FLASH_ACR register.
3. Modify the embedded flash memory clock source and/or the clock prescaler in the RCC_CFGR register of the reset and clock controller (RCC).
4. Check that the new embedded flash memory clock source and/or the new AHB clock prescaler value are taken in account by reading back the embedded flash memory clock source status and/or the prescaler value in the RCC_CFGR register of the reset and clock controller (RCC).
To decrease the CPU frequency:

1. Modify the embedded flash memory clock source and/or the clock prescaler in the RCC_CFRGR register of reset and clock controller (RCC).
2. Check that the embedded flash memory new clock source and/or the new clock prescaler value are taken into account by reading back the embedded flash memory clock source status and/or the AHB interface prescaler value in the RCC_CFRGR register of reset and clock controller (RCC).
3. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in FLASH_ACR register, as described in Table 37.
4. Check that the new number of wait states has been taken into account by reading back the FLASH_ACR register.

**Error code correction (ECC)**

The memory embeds an error correction mechanism. Single-error correction and double-error detection are performed for each read operation. For more details, refer to Section 7.3.8.

**Read errors**

When the ECC mechanism is unable to correct the read operation, the memory reports read errors, as described in Section 7.9.10.

**Read interrupts**

See Section 7.10 for details.

### 7.3.5 FLASH program operations

**Program operation overview**

Program operation consists in issuing write commands. The memory supports the execution of only one write-command at a time. Write-while-write is not supported. Nothing prevents overwriting a non-virgin flash word, but this is not recommended. The result may lead to invalid data and inconsistent ECC code.

**User flash, OBK storage and system flash memories sectors**

For the user and system flash memories, 9-bit ECC is associated to each 128-bit data flash word. In this case, the embedded flash memory must always perform write operations to nonvolatile memory with a 128-bit word granularity. Once the write buffer is full (128 bits), the Busy flag is set, and a programming operation is triggered.

There is a write buffer common to Bank1 and 2, which supports multiple write-access types (128, 64, 32, 16 or 8 bits). The application can decide to write from 8 bits to 128 words. In this case, a force-write mechanism to the 128 bits + ECC is used (see FW bit of FLASH_NS/SECCR register).

When the write request is issued to the memory, any new write request stalls the main AHB bus. Moreover, while a write operation is ongoing, any new read request to the same bank stalls the main AHB bus.
OTP, RO, flash high-cycle data

When the target memory is OTP, RO and flash high-cycle data sectors, 6-bits ECC code is associated to each 16-bit data flash word. The embedded flash memory supports 16- or 32-bit write operations (8-bit write operations are not supported). For 8-bit accesses, write accesses are ignored. There is no write data buffer. Each write access triggers a write in the flash memory.

*Note:* The OTP area is typically write-protected on the final product, as described in Section 7.3.9.

The write protection check is performed at the reception of the write request (during address phase). Write protection is not checked anymore at the output of the write buffer. If a write protection violation is detected, the write operation is canceled, and write protection error (WRPERR) is raised in FLASH_NS/SECSR register.

*Note:* For write protections of main flash, ICP, and OTP see Section 7.6.

Monitoring ongoing write operations

The application software can use a status flag located in FLASH_NS/SECSR to monitor ongoing write operations. Since only one operation is possible at a time, this flag indicates if any operation (write, erase, option change) is ongoing, whatever the bank.

- **BSY:** indicates that an effective write, erase, option byte change, OBK swap, OBK alt sector erase is ongoing in the nonvolatile memory. This flag is not dedicated to a specific bank. It is set when an operation is starting on the memory, whatever the bank. An operation is triggered by:
  - An erase (FLASH_NS/SECCR.STRT)
  - A write (FLASH_NS/SECCR.PG + AHB write)
  - An option modification (FLASH_OPTCR.OPTSTRT)
  - OBKeys sector swap and OBKeys sector erase (FLASH_NS/SECOKCFCGR.SWAP_SECT_REQ and FLASH_NS/SECOKCFCGR.ALT_SECT_ERASE)

They are cleared when the current operation ends, or in case of error.

- **WBNE:** this bit indicates that the embedded flash memory is waiting for new data to complete the 128-bit write buffer. In this state the write buffer is not empty. It is reset as soon as the application software fills the write buffer, or forces the writes by using FW bit in FLASH_NS/SECCR, or an error is detected. When WBNE is high, it is not possible to launch an erase, an option modification, an OBK swap or OBK alternate sector erase operation on flash memory.

- **DBNE:** this bit indicates that the data buffer for parsing 16-bits data is not empty:
  - 16-bit data write access is received, and the data buffer is being filled. It is set at the receipt of the a valid write access, and reset as soon as the write request preparation has been processed.

*Note:* If the memory is busy at the receipt of the AHB write request, the CPU execution is stalled.
### Enabling write operations

Before programming the user flash memory in Bank1 or in Bank2, the application software must ensure that the PG bit is set to 1 in FLASH_NS/SECCR. If not, an unlock sequence must be used (see Section 7.6.7), and the PG bit must be set.

When the option bytes or option bytes key must be modified, or a mass erase must be started, the application software must ensure that FLASH_OPTCR is unlocked. If this is not the case, an unlock sequence must be used (see Section 7.6.7).

A separate mechanism with similar use exists for FLASH_NS/SECOBKCFGR. The control register must be unlocked prior to a start of any OBKeys storage modification. Writing correct sequence to the FLASH_NSOBKKEYR unlocks FLASH_NSOBKCFGR. FLASH_SECSEOBKCFGR is linked to FLASH_SECSEOBKCFGR, as described in the Section 7.6.7.

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

If needed, the application software can update the programming delay, as described in Adjusting programming timing constraints.

### Writing to the FLASH control register FLASH_NS/SECCR and FLASH_OPTCR

The FLASH_NS/SECCR, FLASH_OPTCR and FLASH_NS/SECOBKCFGR registers are not accessible in write mode when the BSY bit is set. Any attempt to write these registers while the BSY bits is set causes the AHB bus to stall until SEC/NSBSY bit is cleared.

#### Single-write sequence

The recommended single-write sequence is the following:

1. Make sure protection mechanism does not prevent programming
2. Check that no flash memory operation is ongoing by checking the BSY bit in the FLASH_NS/SECSR register and CDBNE bits in the FLASH_NS/SECSR register. Check that the write buffer is empty by checking the WBNE bit in the FLASH_NS/SECSR register
3. Check and clear all the error flags due to previous programming/erase operation
4. Unlock the FLASH_NS/SECCR register, as described in Section 7.6.7 (only if this register is not already unlocked)
5. Enable write operations by setting PG bit in the FLASH_NS/SECCR register
6. Write one flash-word at aligned address

**Note:** NS/SECBNE flag indicates if the 128-bit write buffer is waiting for new data.

**Note:** No erase request, options change request, OBK operation is allowed between the first write and the completion of the write operation.

7. Wait for the BSY bit to be cleared in the corresponding FLASH_NS/SECSR register
8. Clear PG bit in FLASH_NS/SECCR register if there are not any more programming requests

If step 6 is executed incrementally (for example byte per byte), the write buffer can become partially filled. In this case the application software can decide to force-write what is stored in the write buffer by using FW bit in FLASH_NS/SECCR register. In this particular case, the unwritten bits are automatically set to 1. If no bit in the write buffer is cleared to 0, the FW bit has no effect.
Note: The usage of a force-write operation prevents the application from updating, in a later stage, the missing bits with a value different from 1. This can lead to unexpected or inconsistent data, or ECC.

Adjusting programming timing constraints

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

The user must therefore trim the optimal programming delay through the WRHIGHFREQ parameter in the FLASH_ACR register. Refer to Table 37 for the recommended programming delay, depending upon the memory clock frequency.

FLASH_ACR configuration register is common to both banks.

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

Caution: Modifying WRHIGHFREQ while programming/erasing the memory can corrupt its content.

Programming errors

When a program operation fails, an error is reported, as described in Section 7.9.

Programming interrupts

See Section 7.10: FLASH interrupts for details.

7.3.6 FLASH erase operations

Erase operation overview

The memory can perform erase operations on 8-Kbyte user sectors, on one user flash memory bank, or on two user flash memory banks (for example mass erase). For more details in user flash memory, ICP, user options and OTP erase protection, see Section 7.6.

Erase commands are issued through the AHB configuration interface. The memory supports one operation at a time, if it receives simultaneously a write and an erase request an error flag is raised, and both operations are canceled. See Section 7.9 for details.

Erase and WRP

If the application software attempts to erase a write-protected user sector, the sector erase operation is aborted, and the WRPERR flag is raised in theFLASH_NS/SECSR register, as described in Section 7.9.2.

Flash busy

Busy signals is described in Monitoring ongoing write operations.

Writing to the FLASH control register FLASH_NS/SECCR and FLASH_OPTCR

Refer to Writing to the FLASH control register FLASH_NS/SECCR and FLASH_OPTCR.
Enabling erase operations

Before erasing a sector, the application software must make sure that FLASH_NS/SECCR is unlocked. If this is not the case, an unlock sequence must be used (see Section 7.6.7).

Note: The application software must not unlock a register that is already unlocked, otherwise this register remains locked until next system reset. This can be used to deliberately lock-out a register from further accesses.

Similar constraints apply to bank erase requests.

Standard flash sector erase sequence

To erase an 8- or 6-Kbyte data user sector without security protections, proceed as follows:

1. Make sure protection mechanism does not prevent sector erase (WRP, secure flag, HDP).
2. Check that no FLASH memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_NSSR register, and that the write buffer is empty by checking the WBNE bit in the FLASH_NSSR register.
3. Check and clear all the non-secure error flags due to previous programming/erase operation. Refer to Section 7.9 for details.
4. Unlock the FLASH_NSCR register, as described in Section 7.6.7 (only if this register is not already unlocked).
5. Set the BKSEL bit, the SER bit and SNB bitfield in the FLASH_NSCR register. BKSEL indicates in which physical bank sector must be erased, then SER indicates a sector erase operation, while SNB contains the target sector number. In case of data sector, use the number of the corresponding regular sector.
6. Set the STRT bit in the FLASH_NSCR register.
7. Wait for the BSY bit to be cleared in the FLASH_NSSR register.
8. STRT bit is automatically cleared at the end of the sector erase, or in case of error.
9. Clear SER in FLASH_NSCR register if there are not anymore sector erase request to be issued.

Note: If another erase flag is requested simultaneously to the sector erase, a PGSERR error is generated.

Secure flash memory sector erase sequence

To erase an 8- or 6-Kbyte data secure configured user sector, proceed as follows:

1. Make sure write protection or HDP mechanism does not prevent sector erase.
2. Ensure that no flash memory operation is ongoing (by checking the BSY and DBNE bits in the FLASH_SECSR register), and that the write buffer is empty (by checking the WBNE bit in the FLASH_SECSR register).
3. Check and clear all the secure error flags due to previous programming/erase operations. Refer to Section 7.9 for details.
4. Unlock the FLASH_SECCR register, as described in Section 7.6.7 (only if this register is not already unlocked).
5. Set the BKSEL bit, the SER bit and SNB bitfield in the FLASH_SECCR register. BKSEL indicates in which physical bank sector must be erased, then SER indicates a sector erase operation, while SNB contains the target secure sector number. In case of data sector, use the number of the corresponding regular sector.
6. Set the STRT bit in the FLASH_SECCR register.
7. Wait for the BSY bit to be cleared in the FLASH_SECSR register.
8. STRT bit is automatically cleared at the end of the sector erase or in case of error.
9. Clear SER in FLASH_SECCR register if there are not any secure sector erase request to be issued.

Note: If another erase flag is requested simultaneously to the sector erase, a PGSERR error is generated.

Standard flash memory bank erase sequence

To erase bank where no sector is configured as secure:
1. Make sure protection mechanism does not prevent sector erase.
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_NSSR register and that the write buffer is empty by checking the WBNE bit in the same register.
3. Check and clear all the non-secure error flags due to previous programming/erase operation. Refer to Section 7.9 for details.
4. Unlock the FLASH_NSCR register, as described in Section 7.6.7 (only if this register is not already unlocked).
5. Set the BKSEL bit and the BER bit in the FLASH_NSCR register to the targeted physical bank (swap setting is ignored).
6. Set the STRT bit in the FLASH_NSCR register to start the bank erase operation. Then wait until the BSY bit is cleared in the FLASH_NSSR register.
7. STRT bit is automatically cleared at the end of erase sequence or in case of error.
8. Clear BER in FLASH_NSCR register if there is not other bank erase request to be issued.

Secure flash memory bank erase sequence

To erase bank where all sectors are configured with secure flag:
1. Make sure protection mechanism does not prevent sector erase (Note that for mass erase the HDP is also considered, it cannot be fully executed from HDPL=2,3 if HDP is defined).
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_SECSR register and that the write buffer is empty by checking the WBNE bit in the same register.
3. Check and clear all the secure error flags due to previous programming/erase operation. Refer to Section 7.9 for details.
4. Unlock the FLASH_SECCR register, as described in Section 7.6.7 (only if this register is not already unlocked).
5. Set the BKSEL bit and the BER bit in the FLASH_SECCR register to the targeted physical bank (swap setting is ignored).
6. Set the STRT bit in the FLASH_SECCR register to start the bank erase operation. Then wait until the BSY bit is cleared in the FLASH_SECSR register.
7. STRT bit is automatically cleared at the end of erase sequence or in case of error.
8. Clear BER in FLASH_SECCR register if there is not other bank erase request to be issued.
Flash mass erase sequence

To erase all sectors of both banks, using non-secure access, all sectors must be configured as non-secure. The application software can set the MER bit to 1 in FLASH_NSCR register, as described below:

1. Make sure protection mechanisms do not prevent mass erase (Section 7.6).
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_NSSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_NSSR register.
3. Check and clear all the non-secure error flags due to previous programming/erase operation. Refer to Section 7.9 for details.
4. Unlock the FLASH_NSCR register as described in Section 7.6.7 (only if the registers are not already unlocked).
5. Set the MER bit to 1 in FLASH_NSCR register.
6. Set the STRT bit in the FLASH_NSCR register. Then wait until BSY bit is cleared in the FLASH_NSSR register.
7. STRT bit is cleared automatically at the end of the erase sequence, or in case of error.

Secure flash memory mass erase sequence

To erase all sectors of both banks simultaneously, using secure access, all sectors must be configured as secure. The application software can set the MER bit to 1 in FLASH_SECCR register, as described below:

1. Make sure protection mechanisms do not prevent mass erase (Section 7.6).
2. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_SECSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_SECSR register.
3. Check and clear all the secure error flags due to previous programming/erase operation. Refer to Section 7.9 for details.
4. Unlock the FLASH_SECCR register as described in Section 7.6.7 (only if the registers are not already unlocked).
5. Set the MER bit to 1 in FLASH_SECCR register.
6. Set the STRT bit in the FLASH_SECCR register. Then wait until BSY bit is cleared in the FLASH_SECSR register.
7. STRT bit is cleared automatically at the end of the erase sequence, or in case of error.

Note: Mass and bank erase also erase high-cycle data sectors aliased from the erased bank.

7.3.7 FLASH parallel operations

As the nonvolatile memory is divided into two independent banks encapsulated in the same macro, the embedded flash memory interface supports a read in one bank while a write (RWW. read while write) or an erase is executed in the other bank. It does not support write-while-write, nor read-while-read. Same is valid for the high-cycle data area, system flash libraries, or the OBK (located on Bank 2).

In all cases, the sequences described in Section 7.3.4, Section 7.3.5 and Section 7.3.6 apply.
7.3.8 FLASH error protections

Error correction codes (ECC)

The embedded flash memory supports an error correction code (ECC) mechanism, based on the SECDED algorithm, to correct single errors and detect double errors.

This mechanism uses nine ECC bits per 128-bit flash word, and applies to user and system memory. For read-only, OTP, flash high-cycle data, a stronger six ECC bits per 16-bit word is used. A double ECC error is generated for an OTP or flash high-cycle data virgin word (for example a word with 22 bits at 1). When this OTP or flash high-cycle data word is no more virgin, the ECC error disappears.

More specifically, during each read operation from a 128-bit flash word, the embedded flash memory retrieves the 9-bit ECC information, computes the ECC of the flash word, and compares the result with the reference value. If they do not match, the corresponding ECC error is raised, as described in Section 7.9.10.

During each program operation, a 9-bit ECC code is associated to each 128-bit data flash word, and the resulting 137-bit flash word information is written in nonvolatile memory.

A similar mechanism applies to read-only and OTP areas, but with 6-bit ECC for 16-bit data.

7.3.9 OTP and RO memory access

OTP and RO memory are accessed through main AHB interface. The OTP is accessible at addresses 0x08FF_F000 to 0x08FF_F7FF, and the read-only section is accessible from 0x08FF_F800 to 0x08FF_FFFF.

FLASH one-time programmable area

The embedded flash memory offers a 2048-byte memory area dedicated to application non-confidential, one-time programmable data (OTP). This area is composed by 1024 words of 16 bits (plus 6 bits of ECC). It cannot be erased, and can be written only once. The OTP area can be accessed through the main AHB interface from address 0x08FF_F000 to 0x08FF_F7FE.

OTP data can be programmed by the application software by 16-bit chunks. Overwriting an already programmed 16-bit half-word can lead to data and ECC errors, and is therefore not supported.

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

When reading OTP data with a single error corrected or a double error detected, the embedded flash memory reports read errors, as described in Section 7.9.10.

When reading OTP data not written by the application software (such as virgin OTP), the ECC correction reports a double-error detection (ECCD), and the data are to be found in the FLASH_ECCDR register. ECCD implies an NMI raised.

OTP write protection

OTP data are organized as 32 blocks of 32 OTP words, as shown in Table 38. An entire OTP block can be protected (locked) from write accesses by setting the LOCKBLi bit (i = 0 to 31) corresponding to each OTP block in the FLASH_OTPBLR option byte register. A block can be write-protected, if it has been programmed (even partially) or not.

The OTP block locking operation is irreversible, and independent from the product life state.
Note: The OTP area can be accessed only in read mode.

Table 38. Flash memory OTP organization

<table>
<thead>
<tr>
<th>OTP block</th>
<th>AHB address</th>
<th>AHB word</th>
<th>Lock bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>[31:16]</td>
<td>[15:0]</td>
</tr>
<tr>
<td>Block 0</td>
<td>0x08FF F000</td>
<td>OTP001</td>
<td>OTP000</td>
</tr>
<tr>
<td></td>
<td>0x08FF F004</td>
<td>OTP003</td>
<td>OTP002</td>
</tr>
<tr>
<td></td>
<td>0x08FF F03C</td>
<td>OTP031</td>
<td>OTP030</td>
</tr>
<tr>
<td>Block 1</td>
<td>0x08FF F040</td>
<td>OTP033</td>
<td>OTP032</td>
</tr>
<tr>
<td></td>
<td>0x08FF F044</td>
<td>OTP035</td>
<td>OTP034</td>
</tr>
<tr>
<td></td>
<td>0x08FF F07C</td>
<td>OTP063</td>
<td>OTP062</td>
</tr>
<tr>
<td>Block 2</td>
<td>0x08FF F080</td>
<td>OTP065</td>
<td>OTP064</td>
</tr>
<tr>
<td></td>
<td>0x08FF F084</td>
<td>OTP067</td>
<td>OTP066</td>
</tr>
<tr>
<td></td>
<td>0x08FF F0BC</td>
<td>OTP95</td>
<td>OTP94</td>
</tr>
<tr>
<td>Block 31</td>
<td>0x08FF F7C0</td>
<td>OTP993</td>
<td>OTP992</td>
</tr>
<tr>
<td></td>
<td>0x08FF F7C4</td>
<td>OTP995</td>
<td>OTP994</td>
</tr>
<tr>
<td></td>
<td>0x08FF F7FC</td>
<td>OTP1023</td>
<td>OTP1022</td>
</tr>
</tbody>
</table>

OTP write sequence

Follow the sequence below to write an OTP word:

1. Check that no flash memory operation is ongoing by checking the BSY bit in the FLASH_NSSR register and that the data buffer is empty by checking the DBNE bit in the FLASH_NSSR register.
2. Check and clear all the error flags due to previous programming/erase operation.
4. Check the protection status of the target OTP word (see Table 38). The corresponding LOCKBLi bit must not be set to 1.
5. Write two OTP words (32 bits) corresponding to the 4-byte aligned address shown in Table 38. Alternatively, the application software can program separately the 16-bit MSB or 16-bit LSB. In this case the first 16-bit write operation starts immediately without waiting for the second one.
6. Wait for the BSY bit to be cleared in the FLASH_NSSR register.
7. Clear PG bit in FLASH_NSCCR register if there is not any programming request anymore in the bank.
8. Optionally, lock the OTP block using LOCKBLi to prevent further data changes.
Note: *Do not write twice an OTP 16-bit word, otherwise an ECC error is generated.*
*Writing OTP data at byte level is not supported, and generates a bus error.*
*To avoid data corruption, it is important to complete the OTP write process (for example by reading back the OTP value), before starting an option change.*

**Flash read-only area**

The embedded flash memory offers a 2-Kbyte area to store read-only data. This area can be accessed through the AHB main port, and is protected by a robust ECC scheme, as detailed in [Section 7.3.8](#).

The read-only information (programmed by STMicroelectronics) that can be used by the application software is detailed in [Table 39](#).

<table>
<thead>
<tr>
<th>Read-only data name</th>
<th>Address</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique device ID</td>
<td>0x08FF F800</td>
<td>U_ID[31:0]</td>
</tr>
<tr>
<td></td>
<td>0x08FF F804</td>
<td>U_ID[63:32]</td>
</tr>
<tr>
<td></td>
<td>0x08FF F808</td>
<td>U_ID[96:64]</td>
</tr>
<tr>
<td>Flash memory size/ package</td>
<td>0x08FF F80C</td>
<td>Flash memory size[15:0] / Package code[15:0]</td>
</tr>
<tr>
<td>Reserved</td>
<td>0x08FF F810 to 0x08FF FFFF</td>
<td>Reserved information</td>
</tr>
</tbody>
</table>

**7.3.10 Flash high-cycle data**

The embedded flash memory offers up to 96 Kbytes (maximum) memory area with high-cycling capability (100 kcycles) to store data and emulate EEPROM. It can be accessed through the AHB system port from address 0x0900_0000 to 0x0901_7FFF (see [Figure 23](#)). It is mapped in the 8 (or 4) last sectors of Bank1 and 2. This area is protected by a robust 6-bit ECC, which allows a read and write granularity of 16 bits at the expense of having sector size shrunk to 6 Kbytes.

A threshold per bank (EDATA(1/2)_STRT) is programmable in user option bytes to determine the beginning of the data flash area. By default, all flash is used for code.

For example, if 48 Kbytes of data are needed in Bank1, set EDATA1_EN to 1, and EDATA1_STRT to 7. If no data are needed in Bank 2, set EDATA2_EN to 0. In this case the data are accessible from address 0x0900_0000 to 0x0900_BFFF for Bank1.

For greater efficiency, it is always recommended to use sector on the other bank for flash high-cycle data, so that the application benefits from RWW capability of the dual-bank arrangement.

When SWAP_BANK feature is enabled, the banks are swapped: the flash high-cycle data in Bank 2 are accessible from 0x0900_0000 to 0x0900_BFFF and the data in Bank 1 are accessible from 0x0900_C000 to 0x0901_7FFF.

A bus error is generated on:
- Attempt to access an address between 0x0900_0000 to 0x0901_7FFF and this address is not valid (EDATA(1/2)_EN not enabled or EDATA(1/2)_STRT not correct).
- Attempt to fetch instructions from flash high-cycle data area.
Erasing the data area sector is possible by normal erase request for corresponding user flash sector (120-127).

Protections and security of high-cycle area are detailed in Section 7.6.9.

Figure 23. Flash high-cycle data memory map on 2 Mbytes device

Note: When flash high-cycle data area on Bank1 is enabled, the code memory map is not continuous from Bank1 to Bank2 on a 2 Mbytes device.

Figure 24. Flash high-cycle data memory map on 1 Mbyte device

7.3.11 Flash bank swapping

Bank1 and Bank2 can be swapped for the user flash. This feature can be used after a firmware upgrade to restart the device on the new firmware. Bank swapping is an user option byte flag controlled by the SWAP_BANK bit of the FLASH_OPTCR register.

Bank specific settings for data area and security attributes follow the original bank and its contents. Control bit BKSEL always refers to physical bank, not the SWAP_BANK setting.

Table 40 shows the accessible memory, depending upon the SWAP_BANK bit value.
Table 40. Memory map and swapping option

<table>
<thead>
<tr>
<th>Flash memory area</th>
<th>Flash memory corresponding bank</th>
<th>Start address</th>
<th>End address</th>
<th>Size (bytes)</th>
<th>Region name</th>
</tr>
</thead>
<tbody>
<tr>
<td>User main memory</td>
<td>SWAP_BANK = 0</td>
<td>0x0800 0000</td>
<td>0x0800 1FFF</td>
<td>8 K</td>
<td>Sector 0</td>
</tr>
<tr>
<td></td>
<td>SWAP_BANK = 1</td>
<td>0x0800 2000</td>
<td>0x0800 3FFF</td>
<td>8 K</td>
<td>Sector 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x080F E000</td>
<td>0x080F FFFF</td>
<td>8 K</td>
<td>Sector 127</td>
</tr>
<tr>
<td></td>
<td>Bank1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bank2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System memory</td>
<td>SWAP_BANK = 0</td>
<td>0x0BF8 0000</td>
<td>0x0BF8 1FFF</td>
<td>8 K</td>
<td>System 1 Sector 0</td>
</tr>
<tr>
<td></td>
<td>SWAP_BANK = 1</td>
<td>0x0BF8 2000</td>
<td>0x0BF8 3FFF</td>
<td>8 K</td>
<td>System 1 Sector 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0BF8 E000</td>
<td>0x0BF8 FFFF</td>
<td>8 K</td>
<td>System 1 Sector 7</td>
</tr>
<tr>
<td></td>
<td>Bank1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bank2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0BF9 0000</td>
<td>0x0BF9 1FFF</td>
<td>8 K</td>
<td>System 2 Sector 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0BF9 2000</td>
<td>0x0BF9 3FFF</td>
<td>8 K</td>
<td>System 2 Sector 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0BF9 E000</td>
<td>0x0BF9 FFFF</td>
<td>8 K</td>
<td>System 2 Sector 7</td>
</tr>
</tbody>
</table>

The SWAP_BANK bit in FLASH_OPTCR register is loaded from the SWAP_BANK option bit only after system reset or POR.

To change the SWAP_BANK bit (for example to apply a new firmware update), follow the sequence below:

1. Check that no flash memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_NS/SECSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_NS/SECSR register.
2. Clear all error flags due to a previous operation.
3. Unlock OPTLOCK bit, if not already unlocked.
4. Set the new desired SWAP_BANK value in the FLASH_OPTSR_PRG register.
5. Start the option byte change sequence by setting the OPTSTRT bit in the FLASH_OPTCR register.
6. Once the option byte change has completed, FLASH_OPTSR_CUR contains the expected SWAP_BANK value, but SWAP_BANK bit in FLASH_OPTCR has not yet been modified and the bank swapping is not yet effective.
7. Force a system reset or a POR. When the reset rises up, the bank swapping is effective (SWAP_BANK value updated in FLASH_OPTCR) and the new firmware shall be executed.
Note: The SWAP_BANK bit in FLASH_OPTCR is read-only, and cannot be modified by the application software.

The SWAP_BANK option bit in FLASH_OPTSR_PRG can be modified whatever the product state. Instead of being locked by PRODUCT_STATE, it is locked by (NS/SEC)BOOT_LOCK User OB.

Figure 25 gives an overview of the bank swapping sequence.

**Figure 25. Flash bank swapping sequence**

### 7.3.12 FLASH reset and clocks

**Reset management**

The embedded flash memory can be reset by a core domain reset, driven by the reset and clock control (RCC). The main effects of this reset are the following:

- All registers, except for option byte registers, are cleared, including read and write latencies. If the bank swapping option is changed, it is applied.
- Most control registers are automatically protected against write operations. To unprotect them, new unlock sequences must be used as described in Section 7.6.7.

The memory can be reset by a power-on core domain reset, driven by the reset and clock control (RCC). When the reset falls, all option byte registers are reset. When the reset rises up, the option bytes are loaded, potentially applying new features. During this loading sequence, the device remains under reset, and the memory is not accessible.

The reset signal can have a critical impact on the memory: the content is not guaranteed if a device reset occurs during a write or erase operation.
Reset occurring during flash operation

If a reset occurs during a flash operation (programming, erase or option change), the content of the memory is not guaranteed. It is mandatory for integrity to restart the operation. The status register FLASH_OPSR gives information about operations interrupted by a reset.

FLASH_OPSR.CODE_OP gives opcode of operation. Table 41 indicates how to use FLASH_OPSR, and which operation is required.

Table 41. Recommended reactions to FLASH_OPSR contents

<table>
<thead>
<tr>
<th>CODE_OP</th>
<th>Operation interrupted</th>
<th>OTP_OP</th>
<th>SYSF_OP</th>
<th>BK_OP</th>
<th>DATA_OP</th>
<th>Flash area</th>
<th>ADDR_OP min</th>
<th>ADDR_OP max</th>
<th>Recommended action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>No operation ongoing while reset</td>
<td>0 0 0 0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>ADDR_OP min</td>
<td>-</td>
<td>-</td>
<td>No extra action</td>
</tr>
<tr>
<td>0x001</td>
<td>Write operation</td>
<td>0 0 0/1 0</td>
<td>User flash</td>
<td>0x0000</td>
<td>0xFFFF</td>
<td>0 1 0/1 0</td>
<td>System flash</td>
<td>0x0000</td>
<td>0x0FFF</td>
</tr>
<tr>
<td></td>
<td>1 0 0 0</td>
<td>OTP</td>
<td>0x0600</td>
<td>0x07FF</td>
<td>1 0 1 0</td>
<td>OBKeys(2)</td>
<td>0x0000</td>
<td>0x03FF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 0 0/1 1</td>
<td>Data area(3)</td>
<td>0xF000</td>
<td>0xFFFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>OBK alternate sector erase</td>
<td>0 0 1 0</td>
<td>OBKeys</td>
<td>0</td>
<td>0</td>
<td>Relaunch the alternate sector erase</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x011</td>
<td>Sector erase(4)</td>
<td>0 0 0/1 0</td>
<td>User flash</td>
<td>0x0000</td>
<td>0xFFFF</td>
<td>0 1 0/1 0</td>
<td>System flash</td>
<td>0x0000</td>
<td>0x0FFF</td>
</tr>
<tr>
<td>0x100</td>
<td>Bank erase</td>
<td>0 0 0/1 0</td>
<td>User flash</td>
<td>-</td>
<td>-</td>
<td>Relaunch bank erase</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x101</td>
<td>Mass erase</td>
<td>0 0 0 0</td>
<td>-</td>
<td>-</td>
<td>Relaunch mass erase</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x110</td>
<td>Option change</td>
<td>0 0 0 0</td>
<td>User configuration</td>
<td>-</td>
<td>-</td>
<td>New attempt on option change</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x111</td>
<td>OBK swap sector</td>
<td>0 0 1 0</td>
<td>OBKeys</td>
<td>-</td>
<td>-</td>
<td>Erase alternate sector, reprogram new OBKeys and relaunch swap</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Dash represents “does not matter”.
2. Depends on current OBK sector.
3. Depends on EDATA setting in the OB.
4. Addresses indicated are aligned to sector start, data area sectors are erased by erase request to corresponding user flash memory sector.

Clock management

The memory uses the microcontroller system clock (sys_ck), here the AHB interface clock.
7.4 FLASH option bytes

7.4.1 About option bytes

The memory includes a set of nonvolatile option bytes. They are loaded at power-on reset and can be read and modified only through configuration registers. This section documents:

- when option bytes are loaded
- how application software can modify them
- the detailed list of option bytes, together with their initial values (before the first option byte change, user default configuration).

7.4.2 Option-byte loading

There are multiple ways of loading the option bytes into embedded flash memory:

- **Power-on wakeup**
  When the device is first powered, the embedded flash memory automatically loads all the option bytes. During the option byte loading sequence, the device remains under reset and the embedded flash memory cannot be accessed.

- **Wakeup from system Standby**
  When the core power domain, which contains the embedded flash memory, is switched from Standby mode to Run mode, the embedded flash memory behaves as during a power-on sequence. During loading time the device is not under reset, unlike power-on sequence.

- **Dedicated option byte reloading by the application**
  When the user application successfully modifies the option byte content through the memory registers, the nonvolatile option bytes are programmed and the memory automatically reloads all option bytes to update the option registers.

*Note:* The option-byte read sequence is protected by error correction code. In case of error, the option bytes are loaded with default values (see Section 7.4.3), different from the initial values (user default configuration), and more restrictive.

7.4.3 Option-byte modification

**Changing user option bytes**

A user option-byte change operation can be used to modify the configuration and the protection settings saved in the nonvolatile option byte area.

There are several rules enforced when attempting to change a user option byte, they are summarized in Section 7.4.7. Failing to stick to those rules usually results in errors, described in Section 7.9.12.

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

- The first register set contains the current values of the option bytes. Their names have the _CUR extension. All “_CUR” registers are read-only. Their values are automatically loaded from the nonvolatile memory after power-on reset, wakeup from system standby or after an option byte change operation.

- The second register set allows the modification of the option bytes. Their names contain the _PRG extension. All “_PRG” registers can be accessed in read/write mode.
When the OPTLOCK bit in FLASH_OPTCR register is set, modifying the FLASH_XXX_PRG registers is not possible.

When OPTSTRT bit is set to 1, the memory checks the programming sequence (PGSERR) and the conditions described in Section 7.4.7 (OPTCHANGEERR). If no error has been detected (PGSERR/OPTCHANGEERR), the embedded flash memory launches the option byte modification in its nonvolatile memory and updates the option byte registers with _CUR extension.

If one of the condition described in Section 7.4.7, Section 7.9.12 or Section 7.9.5, alternatively Section 7.9.4 is not respected, the memory aborts the option byte change operation. In this case, the FLASH_XXX_PRG registers are not overwritten by current option value. The user application can check what was wrong in their configuration.

Unlocking the option-byte modification

After reset, the OPTLOCK bit is set to 1 and the FLASH_OPTCR is locked. As a result, the application software must unlock the option configuration register before attempting to change the option bytes. The FLASH_OPTCR unlock sequence is described in Section 7.6.7.

Option-byte modification sequence

To modify user option bytes, follow the sequence below:

1. Check that no FLASH memory operation is ongoing by checking the BSY bit in the FLASH_NS/SECSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_NS/SECSR register.
2. Check the data buffer is empty (DBNE = 0) in FLASH_NS/SECSR register.
3. Clear all error flags due to a previous operation.
4. Unlock FLASH_OPTCR register as described in Section 7.6.7, unless the register is already unlocked.
5. Write the desired new option byte values in the corresponding option registers (FLASH_XXX_PRG).
6. Set the option byte start change OPTSTRT bit to 1 in the FLASH_OPTCR register.
7. Wait until BSY bit is cleared in FLASH_NS/SECSR register.
8. OPTSTRT bit is cleared automatically at the end of the sequence (or in case of error).
9. Reset the device. This step is always recommended, becomes mandatory when one of the following option bytes is impacted:
   a) SECBOOTADDR
   b) NSBOOTADDR
   c) TZEN
   d) BOOT_UBE

Note: If a reset or a power-down occurs while the option byte modification is ongoing, the original option byte value is kept. A new option byte modification sequence is required to program it.
Option-byte overview

*Table 42* lists all the user option bytes managed through the memory registers, as well as initial values before the first option byte change (user default configuration).

| Register           | Bits 31  | Bits 30  | Bits 29  | Bits 28  | Bits 27  | Bits 26  | Bits 25  | Bits 24  | Bits 23  | Bits 22  | Bits 21  | Bits 20  | Bits 19  | Bits 18  | Bits 17  | Bits 16  | Bits 15  | Bits 14  | Bits 13  | Bits 12  | Bits 11  | Bits 10  | Bits 9   | Bits 8   | Bits 7   | Bits 6   | Bits 5   | Bits 4   | Bits 3   | Bits 2   | Bits 1   | Bits 0   |
|--------------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
|                   | 0       | 0       | 1       | 0       | 1       | 0       | 1       | 0       | 1       | 0       | 1       | 0       | 0       | 0       | 1       | 1       | 1       | 0       | 1       | 1       | 1       | 0       | 1       | 1       | 1       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       |
|                   | 1       | 1       | 0       | 0       | 0       | 1       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 1       | 1       | 1       | 1       | 1       | 1       | 0       | 0       | 0       |
| FLASH_BOOTR       | SECBOOTADD[23:8]| SECBOOTADD[7:0]| SECBOOT_LOCK| 0xC000| 0x000| 0xC3|
| FLASH_NSBOOTR     | NSBOOTADD[23:8]| NSBOOTADD[7:0]| NSBOOT_LOCK| 0x0800| 0x000| 0xC3|
|                   | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       |
|                   | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       |
| FLASH_WRPSGN1R    | WRPSG[127:124]| WRPSG[123:120]| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| |
|                   | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       | 1       |
| FLASH_WRPSGN2R    | WRPSG[27:124]| WRPSG[23:120]| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| ...| |
### 7.4.4 Description of user and system option bytes

The general-purpose option bytes that can be used by the application are listed below:

- **Watchdog management**
  - **IWDG_STOP**: independent watchdog IWDG (also known as WDGLS_CD) counter active in Stop mode if 1 (stop counting or freeze if 0)
  - **IWDG_STDBY**: independent watchdog IWDG (also known as WDGLS_CD) counter active in Standby mode if 1 (stop counting or freeze if 0)
  - **IWDG_SW**: hardware (0) or software (1) IWDG (also known as WDGLS_CD) watchdog control selection

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

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

---

### Table 42. Option-byte organization (continued)

| Register            | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| FLASH_OTPBLR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| LOCKBL              | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|                     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|                     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|                     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  |
|                     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|                     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|                     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
When the IWDG is kept running during Stop or Standby mode, it can wake up the device from these modes.

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

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

- **Device options**
  - IO_VDDIO2_HSLV: enables the configuration of pads below 2.7 V for VDDIO2 power rail if set to 1
  - IO_VDD_HSLV: enables the configuration of pads below 2.7 V for VDD power rail if set to 1
  - USBPD_DIS: bit to disable USB PD

At device deliver, the values programmed in the general-purpose option bytes are the following:

- **Watchdog management**
  - IWDG (also known as WDGLS_CD) active in Standby and Stop modes: 0x1
  - IWDG (also known as WDGLS_CD) not automatically enabled at power-on: 0x1

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

- **Device working in the full voltage range with I/O speed optimization at low-voltage disabled (IO_VDDIO2_HSLV = 0 and IO_VDD_HSLV = 0)**

Refer to **Section 7.11** for details.

### 7.4.5 Description of data protection option bytes

The option bytes that can be used to enhance data protection are listed below:

- **PRODUCT_STATE[7:0]:** A product life cycle state (see **Section 7.6.11** for details).
- **WRPSGn1/2:** write protection option of the corresponding group of four consecutive sectors in Bank1 (respectively Bank2). It is active low. Refer to **Section 7.6.8** for details.
  - Bit N: Group embedding sectors 4 x N to 4 x N + 3
- **SECWMx:** TrustZone® secure only watermark area definition (refer to **Section 7.6.1** for details).
  - SECWM1_STRT (respectively SECWM1_END) contains the first (respectively last) sector of the secure access only zone in Bank1
  - SECWM2_STRT (respectively SECWM1_END) contains the first (respectively last) sector of the secure access only zone in Bank 2.
- **TZEN:** this nonvolatile option can be used by the application to activate the secure access mode, as described in **Section 7.6.** For security reasons the TZEN is stored with redundancy on 8 bits. While the TZEN OB value is set immediately after
programming, to actually use TZ features, reset is required to set the TZ_STATE in SBS.

When TZEN is activated, secure watermark settings and secure boot address may be reset to default to prevent a deadlock in case the previous values were not correct.

- HDPx: Secure hide protection - HDPL exclusive area control in the user flash memory.

When factory programmed values of the data protection option bytes are the following:

- Product state depends on sales type
- Flash bank erase operations do not impact watermarked secure data areas
- Secure watermark areas protections disabled (start addresses higher than end addresses)
- Write protection disabled (all option byte bits set to 1)
- TrustZone Secure access mode disabled (TZEN option byte value = 0xC3)

Refer to Section 7.11 for details.

### 7.4.6 Description of boot address option bytes

Below the list of option bytes that can be used to configure the appropriate boot address for an application:

- **PRODUCT_STATE**
- **BOOT_UBE**: Available only on cryptography enabled devices. Selects either ST-iRoT or User flash as default boot address. Also used by RSS(SFI) to choose either Bootloader or ST-iRoT as next stage.
- **NSBOOTADD**: Selects default boot address when TZ is disabled.
- **SECBOOTADD**: Selects default boot address when TZ is enabled.
- (NS/SEC)BOOT_LOCK: Protects the boot configuration from further modification attempts.
- **SWAP_BANK**: bank swapping option, set to 1 to swap user flash banks after boot (see Section 7.3.11). If BOOT_LOCK corresponding to TZEN state (SEC/NS) is active, the value in SWAP_BANK is fixed, read-only.

When STMicroelectronics delivers the device, the PRODUCT_STATE is Open, BOOT_LOCK is not set (0xC3) and the BOOT_UBE is set to user flash (0xB4). Addresses are SECBOOTADD = 0x0C00 0000, NSBOOTADD = 0x0800 0000.

Refer to Section 7.11 for details.

### 7.4.7 Specific rules for modifying option bytes

On top of OPTLOCK bit and register access rules, there are also other protection means for selected security-sensitive option byte fields.

More option bytes can be modified simultaneously, but if they rely on each other for protection, both states are checked. For example, when trying to modify PRODUCT_STATE and TZEN simultaneously, resulting state must be coherent with the rules. With few exceptions, listed in this section, failing to uphold the rules results in raising OPTCHANGEERR flag (Section 7.9.12 for additional details).
Even in the closed PRODUCT_STATE progression, some OB can still be modified, if all the other constraints are satisfied. An overview is presented in Table 44.

### Table 43. Specific OB modifying rules

<table>
<thead>
<tr>
<th>Option byte</th>
<th>HDPL</th>
<th>TZ secure</th>
<th>Value</th>
<th>Product state</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT_STATE</td>
<td>0,1(1)</td>
<td>Some values only when TZ is enabled</td>
<td>Set of possible transitions</td>
<td>Set of possible transitions (Table 37)</td>
</tr>
<tr>
<td>HDP</td>
<td>0,1(2)</td>
<td>(3)</td>
<td>-</td>
<td>Refer to Table 44</td>
</tr>
<tr>
<td>TZEN</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Regression, Open or Provisioning</td>
</tr>
<tr>
<td>EPOCH</td>
<td>0</td>
<td>-</td>
<td>PRG &gt; CUR</td>
<td>Regression</td>
</tr>
<tr>
<td>EPOCH_NS</td>
<td>0(2)</td>
<td>-</td>
<td>PRG &gt; CUR</td>
<td>NS-Regression, Regression</td>
</tr>
<tr>
<td>SECWM</td>
<td>-</td>
<td>Secure mode only(2)</td>
<td>-</td>
<td>Refer to Table 44</td>
</tr>
<tr>
<td>BOOT_UBE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Open or Provisioning, SECBOOT_LOCK disabled</td>
</tr>
<tr>
<td>SECBOOTADD</td>
<td>-</td>
<td>Secure mode only(2)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>NSBOOTADD</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Open or Provisioning, NSBOOT_LOCK disabled</td>
</tr>
<tr>
<td>LOCKBL</td>
<td>-</td>
<td>-</td>
<td>One way switch(2)</td>
<td></td>
</tr>
<tr>
<td>SWAP_BANK</td>
<td>-</td>
<td>Fixed if BOOT_LOCK corresponding to TZEN is set</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SECBOOT_LOCK</td>
<td>-</td>
<td>Secure mode only(2)</td>
<td>-</td>
<td>Open, Regression or Provisioning to unlock</td>
</tr>
<tr>
<td>NSBOOT_LOCK</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

1. Most transitions are possible regardless of HDPL, only a few require specific HDPL, see text below.
2. No OPTCHANGEERR raised in violation of this.
3. Dash means there is no limitation from this side.

### Table 44. OB modifiable in closed product

<table>
<thead>
<tr>
<th>PRODUCT_STATE</th>
<th>OBs that may be modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>TZ-Closed</td>
<td>SWAP_BANK, LOCKBL, PRODUCT_STATE, SECWM, HDP</td>
</tr>
<tr>
<td>Closed</td>
<td>SWAP_BANK, LOCKBL, PRODUCT_STATE</td>
</tr>
<tr>
<td>Locked</td>
<td>SWAP_BANK, LOCKBL</td>
</tr>
</tbody>
</table>

Specific rules must be respected to update the following OB:

- **PRODUCT_STATE**
  
  PRODUCT_STATE transitions follow a state machine with two types of transition. Locking down the product is allowed without restriction. Opening the product is possible only using a debug interface and following a digital signature verification. The regression to a less secure state results in the erasing of the protected content. More details are given in Section 7.6.11.
  
  Summary: Selected changes are only possible in HDPL1, regression in HDPL0. Some states are accessible only when TZEN is enabled.
• **HDP**
  Can only be modified in HDPL0 and HDPL1. Must not be set to overlap with flash high-cycle data area.

• **TrustZone access mode (TZEN)**
  Can only be changed in Open, Regression and Provisioning.

• **SEC_EPOCH and NS_EPOCH**
  The value programmed must be greater than current value. The increment is done in specific product states: Regression and NS-Regression.

• **Secure watermark area (SECWM1/2_STRT and SECWM1/2_END)**
  Can be changed only in secure mode (TZ_state = 0xB4). Automatically reset to default value when TZEN is enabled. Must not be set to overlap with flash high-cycle data area.

• **BOOT_UBE**
  Available only on cryptography enabled devices. Can only be changed in product states open for debug like Open and also in Provisioning. SECBOOT_LOCK must be disabled to change.

• **SECBOOTADD**
  Can only be changed in Open and Provisioning. Locked by SECBOOT_LOCK. Automatically reset to default value when TZEN is enabled.

• **NSBOOTADD**
  Can only be changed in Open and Provisioning. Locked by NSBOOT_LOCK.

• **LOCKBL**
  Can only be changed freely in one direction. A permanent irreversible switch.

• **SWAP_BANK**
  Not modifiable when both TZEN and SECBOOT_LOCK are 0xB4 (set) or when TZEN = 0xC3 (disabled) and NSBOOT_LOCK is active (0xB4).

• **SECBOOT_LOCK**
  Can only be changed freely in locking direction. Unlock is possible in Open, Provisioning and Regression.

• **NSBOOT_LOCK**
  Can only be changed freely in locking direction. Unlock is possible in Open, Provisioning and Regression.

**Note:** For all user option bytes above: default values are loaded and Tamper is signaled when a double ECC error occurs during OBL.

### 7.5 Option-byte key management

This section describes the key storage tied with SAES and OBK-HDPL value set by the SBS (refer to Section 14.3.7: SBS hardware secure storage control). The OBKs are stored in 128 bits and protected by 9-bit ECC (SEC/DED). They are written alternatively in two sectors.

#### 7.5.1 OBK loading

OBKs are memory mapped, and accessible through the main AHB bus (C-Bus), starting from address 0x0FFD 0000. A 9-bit ECC is associated to each 128-bit data flash word.
7.5.2 OBK access per HDPL level

Table 45 describes the option-byte key areas. The key storage is not dedicated to a particular key, and the usage is defined by the application.

An option-byte key can be accessed only if the OBK-HDPL (set in SBS) matches the HDPL associated to the storage offset (as indicated below) (refer to Section 14.3.7: SBS hardware secure storage control). If it is not the case, an OBKERR error is raised.

<table>
<thead>
<tr>
<th>OBK-HDPL</th>
<th>Address offset start</th>
<th>Address offset end</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x0000</td>
<td>0x00FF</td>
<td>Reserved for HDPL0 key storage</td>
</tr>
<tr>
<td>1</td>
<td>0x0100</td>
<td>0x08FF</td>
<td>OEM iRoT keys</td>
</tr>
<tr>
<td>2</td>
<td>0x0900</td>
<td>0x0BFF</td>
<td>uRoT, OS or secure application</td>
</tr>
<tr>
<td>3</td>
<td>0x0C00</td>
<td>0x17FF</td>
<td>HDPL 3 secure keys</td>
</tr>
<tr>
<td>3</td>
<td>0x1800</td>
<td>0x1FEF</td>
<td>HDPL 3 non-secure keys</td>
</tr>
</tbody>
</table>

7.5.3 OBK programming sequence

The sequence to write OBK is the same as a write to main flash except that the “alternate sector” (ALT_SECT) bit must be set to write in the alternate OBK sector. The ALT_SECT bit is checked each time a write access is received. If it does not match the one stored in the write buffer, an error is raised.

Note: Before writing in the alternate OBK sector the user must ensure that the whole key space (128, 256, or 512 bits) is free, by reading the alternate OBK sector.

If the alternate sector is not free, the user must erase the sector using the dedicated erase command ALT_SECT_ERASE.

Secure programming method

Using the secure register if TrustZone is active (TZ_STATE = 0xB4):

1. Program the ALT_SECT bit.
2. Make sure protection mechanism does not prevent programming (TZ, PRIV and OBK-HDPL).
3. Check that no FLASH memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_SECSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_SECSR register.
4. Check and clear all the secure error flags due to previous operation.
5. Unlock the FLASH_SECCR register only if this register is not already unlocked, as described in Section 7.6.7.
6. Enable write operations by setting the SECPG bit in the FLASH_SECCR register.
7. Write one flash-word corresponding to 16-byte data starting at 16-byte aligned address.

Note: WBNE flag indicates if the 128-bit write buffer is waiting for new data.
Note: No erase request, options change request, OBK swap sector or OBK alternate sector erase is allowed between first write and the completion of flash write operation.

8. Wait for the BSY bit to be cleared in the corresponding FLASH_SECSR register.
9. Clear PG bit in FLASH_SECCR register if there is not any programming request anymore in the bank.

If step 7 is executed incrementally (byte per byte), the write buffer can become partially filled. In this case the application software can decide to force-write what is stored in the write buffer by using FW bit in FLASH_SECCR register. In this particular case, the unwritten bits are automatically set to 1. If no bit in the write buffer is cleared to 0, the FW bit has no effect.
7.5.4 OBK programming finite state machine

When the OBK write buffer is filled, the OBK programing is launched automatically either in the current OBK sector or in the alternate OBK sector depending on the ALT SECT bit.

If the address is not virgin, the OBKWRERR is raised.
7.5.5 OBK swap sector

The user can request a swap of the Option Byte Keys by setting the bit SWAP_SECT_REQ in the NS/SECOBKCFGR register. The number of keys swapped are defined by the SWAP_OFFSET in NS/SECOBKCFGR register.

Swap is limited by OBK-HDPL value, following SWAP_OFFSET values are permitted:

- OBK_HDPL0_OFFSET = 16
- OBK_HDPL1_OFFSET = 144
- OBK_HDPL2_OFFSET = 192
- OBK_HDPL3SEC_OFFSET = 384
- OBK_HDPL3NS_OFFSET = 511
The programming sequence is the following:

1. Check that no FLASH memory operation is ongoing by checking the BSY and DBNE bit in the FLASH_SEC/NSSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_SEC/NSSR register.

2. Check and clear all the secure/non-secure error flags due to previous operation.

3. Unlock NS/SECOBKCFGR register if not unlocked already thanks to FLASH_NS/SECOBKKEYR.

4. Define the numbers of keys to be swapped thanks to SWAP_OFFSET field in NS/SECOBKCFGR. SWAP_OFFSET must be equal to or greater than OBK_HDPL<\text{N-1}>_OFFSET.

5. Set bit SWAP_SECT_REQ in NS/SECOBKCFGR.

6. Wait for BSY bit to be cleared. In case of error OBKERR flag is set.

7. SWAP_SECT_REQ is cleared automatically at the end of the operation or in case of error.
While the OBK SWAP operation is ongoing, the BSY flag is set and no other operation (write, erase, user opt change) can be launched in parallel.
7.5.6 OBK alternate sector erase

The user can request the erase of the alternate OBK sector by setting the bit ALT_SECT_ERASE in the FLASH_NS/SECOBKCFGR register.

The programming sequence must be:
1. Check that no FLASH memory operation is ongoing by checking the BSY and DBNE bits in the FLASH_SEC/NSSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_SEC/NSSR register.
2. Check and clear all the secure/non-secure error flags due to previous operation.
3. Set bit ALT_SECT_ERASE in NS/SECOBKCFGR.
4. Wait for BSY bit to be cleared.
5. ALT_SECT_ERASE is cleared automatically at the end of the operation or in case of error.

7.6 FLASH security and protections

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

For that purpose FLASH implements the following protection mechanisms:
- TrustZone backed watermark and block security protection
- Temporal isolation protection (HDP)
- Configuration protection
- User flash write protection
- Device nonvolatile security life cycle and application boot state management
- OTP locking

The microcontroller can have TrustZone active (TZ_STATE = 0xB4) or not (TZ_STATE = 0xC3).

While TrustZone is active the flash memory can be accessed in four basic modes:
- non-secure and unprivilege
- non-secure and privilege
- secure and privilege
- secure and unprivilege

While TrustZone is disabled, only two modes are possible:
- non-secure and unprivilege
- non-secure and privilege

The flash interface evaluates the access restrictions in the following order:
1. TrustZone security
2. Write protection (for write access)
3. HDP
4. Privilege
7.6.1 TrustZone security protection

The global TrustZone system security is activated by setting the TZEN option in the FLASH_OPTSR2_PRG register. The actual state of TZ is however determined by the SBS and the setting of TZ_STATE (refer to Section 14.3.5: SBS boot control).

When TrustZone is active (TZ_STATE = 0xB4), the following additional security features are available:

- Secure watermark-based user options bytes defining secure areas
- Secure or non-secure block-based areas can be configured on-the-fly after reset. This is a volatile secure area
- Additional product states associated with closed_secure provide protection of secure domain against external access (debug or bootloader)
- Erase or program operation can be performed in secure or non-secure mode with associated configuration bit. When the TrustZone is disabled (TZ_STATE = 0xC3), the above features are deactivated and all secure registers are RAZ/WI.

Activating TrustZone security

When the TrustZone is activated (TZ_STATE = 0xB4), the SECBOOTADD and the secure watermark areas are set to a secure default value. Default settings are also used in case of corrupted configuration. In this default state, user flash memory is secure as shown in the table below.

<table>
<thead>
<tr>
<th>Table 46. Default secure watermark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure watermark in case of configuration error</td>
</tr>
<tr>
<td>SECWMx_STRT = 0</td>
</tr>
</tbody>
</table>

Illegal access generation

A non-secure access to a secure flash memory area is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the FLASHIE illegal access interrupt is enabled in the TZIC_IER2 register.

A non-secure access to a secure flash register generates an illegal access event. An illegal access interrupt is generated if the FLASH_REGIE illegal access interrupt is enabled in the TZIC_IER2 register.

<table>
<thead>
<tr>
<th>Table 47. Flash memory TZ protection summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>TZ protection (TZ_STATE = 0xB4)</td>
</tr>
<tr>
<td>Secure</td>
</tr>
<tr>
<td>Fetch</td>
</tr>
<tr>
<td>Read</td>
</tr>
<tr>
<td>Write</td>
</tr>
</tbody>
</table>
Deactivate TrustZone security

Reversing of the TZ setting in OB (TZEN) is possible when the product state is ST-RoT-Ready or Provisioning. Deactivating TZEN from other states requires full regression (Regression state).

When the TrustZone is deactivated (TZ_STATE = 0xC3) after option bytes loading, the following security features are deactivated:

- Watermark-based secure area (refer to Watermark-based secure flash area protection)
- block-based secure area (refer to Section 7.6.3)
- Secure interrupts (refer to Section 7.9.11)
- All secure registers are RAZ/WI.

Watermark-based secure flash area protection

When TrustZone security is active (TZ_STATE = 0xB4), a part of the flash memory can be protected against non-secure read and write accesses. Up to two different nonvolatile secure areas can be defined by option bytes and can be read or written by a secure access only: one area per bank can be selected with a sector granularity.

The secure areas are defined by a start sector offset and end sector offset using the SECWMx_STRT and SECWMx_END option bytes. These offsets are defined in the secure watermark registers address registers FLASH security watermark for Bank 1 (FLASH_SECWM1R_CUR), FLASH security watermark for Bank 2 (FLASH_SECWM2R_CUR) and modified by corresponding FLASH_XXX_PRG registers.

The SECWMx_STRT and SECWMx_END option bytes can only be modified by secure firmware.
Caution: Switching a memory area from secure to no-secure does not erase its content. The user secure software must perform the needed operation to erase the secure area before switching an area to non-secure attribute whenever is needed. It is also recommended to flush the instruction cache.

7.6.2 Hide protection (HDP)

The HDP area is independent of the flash watermark-based secure area. One nonvolatile secure hide protection (HDP) area per bank can be defined with a sector granularity. Access to the hide protection area can be denied by progressing the HDPL level in the SBS.

When the HDPL = 1, no user flash is protected by the HDP. With HDPL ≥ 2 the user OB defined HDP area in each bank is closed. No read, write, fetch or erase is allowed in the HDP area.

The HDP area can be extended, the extension setting is only a register value, not stored in OB. With HDPL = 3 the HDP area remains activated and the extension is added. If extension is added while the base user OB defined area is not active, the extension covers one more sector, because HDPx_END sector is also protected.

The HDPL level can be only cleared by a system reset, there is no means to deactivate the HDP area.

The protected HDP area is defined by setting its size using start and end sectors in a similar way as the secure watermark.

The size of the inaccessible area can be extended using register FLASH_HDPEXTR values HDPx_EXT. The value HDPx_EXT is a number of sectors added to the HDP area (past the HDPx_END sector). The volatile HDPx_EXT value in the register can be increased, but cannot decrement.

In case of HDPx_STRT > HDPx_END the extension will cover (HDPx_EXT + 1) size area between sectors marked by HDPx_END and HDPx_END + HDPx_EXT.

By default both the HDPx_END is set to zero and HDPx_STRT is set to one. This means that the HDP area in user flash is zero size and extends from the flash start address (first sector).
For example, to protect area by HDP from the address 0x0801 4000 (included) to the address 0x0804 FFFF (included):

- For physical Bank1, the option-byte registers must be programmed with:
  - HDP1_STRT = 0x0A
  - HDP1_END = 0x27

Figure 29. HDP in user flash memory

- Once the HDPL is incremented to 2, the marked area becomes inaccessible. The HDPL = 2 code then sets the extension size to 29. Once the HDPL is incremented to 3, the extension area becomes also inaccessible.

- Alternatively, if:
  
  HDP1_END = 0  
  HDP1_START = 1  
  HDPx_EXT = 1

  Sector 0 and Sector 1 are HDP protected.

  Or a rather unusual example:

  HDP1_END = 2  
  HDP1 START = 7  
  HDPx_EXT = 3

  Sectors 2 to 5 are HDP protected.

  Here the user should know that if his application needs sectors 2 to 4 only as HDP area, he should program HDPx_EXT at 2 instead of 3.

- If the two banks are swapped, the protection defined to physical Bank1 remains on the physical Bank1, unaffected by swapping. Separate protection applies to physical Bank2 and the option bytes registers must also be programmed with:
  - HDP2_STRT = 0x0A
  - HDP2_END = 0x27

Note: For more details on the bank swapping mechanism, refer to Section 7.6.6.

Table 50. Secure hide protection

<table>
<thead>
<tr>
<th>HDPx watermark option-byte values (x = 1, 2)</th>
<th>Hide protection area</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDPL ≤ 1</td>
<td>No inaccessible HDP area in user flash memory</td>
</tr>
<tr>
<td>HDPL = 2</td>
<td></td>
</tr>
<tr>
<td>HDPx_END &lt; HDPx_STRT</td>
<td>No inaccessible HDP area in user flash memory</td>
</tr>
<tr>
<td>HDPx_END = HDPx_STRT</td>
<td>Exactly one sector is protected by HDP.</td>
</tr>
<tr>
<td>HDPx_END &gt; HDPx_STRT</td>
<td>The area between HDPx_STRT and HDPx_END is HDP protected.</td>
</tr>
</tbody>
</table>
7.6.3 Block-based secure flash memory area protection

Any sector can be programmed on-the-fly as secure or non-secure using the block-based configuration registers. FLASH_SECBB1x (respectively FLASH_SECBB2x) registers are used to configure the security attribute for sectors in Bank1 (respectively Bank2).

When the sector security attribute bit SECBByx[i] is set, the security attribute is the same as the secure watermark-based area. The secure sector is only accessible by a secure access.

If the SECBByx[i] bit is set or reset for a sector already included in a secure watermark-based area, the sector keeps the watermark-based protection security attributes.

To modify a block-based sector security attribute, it is recommended to:
- Check that no flash operation is ongoing on the related sector.
- Add an ISB instruction after modifying the sector security attribute SECBByx[i].

**Caution:** Switching a sector from secure to non-secure does not erase the content of the associated sector. User secure software must perform the needed operations before switching to non-secure attribute:
- Erase sector content.
- Invalidate the instruction cache.

**Note:** For SECBByx[i] access control, refer to Table 52.

Execute protection for S-Bus (AHB register bus) is done in AHB decoder, not in flash memory interface itself.
### 7.6.4 Block-based privileged flash memory area protection

Any sector can be programmed on-the-fly as privileged or unprivileged using the block-based configuration registers. FLASH_PRIVBB1x (respectively FLASH_PRIVBB2x) registers are used to configure the privilege attribute for sectors in Bank1 (respectively Bank2).

When the sector privilege attribute PRIVBByx[i] bit is set, the sector is only accessible by a privileged access. An unprivileged sector is accessible by a privileged or unprivileged access.

To modify a block-based privilege attribution, it is recommended to:
- Check that no flash operation is ongoing on the related sector.
- Add an ISB instruction after modifying the sector security attribute PRIVBByx[i].

**Caution:** Switching a sector from privileged to unprivileged does not erase the content of the associated sector.

<table>
<thead>
<tr>
<th>Access (TZ not relevant)</th>
<th>Unprivileged sector</th>
<th>Privileged sector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privilege</td>
<td>Fetch</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Read</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Erase</td>
<td></td>
</tr>
<tr>
<td>Unprivilege</td>
<td>Fetch</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>Read</td>
<td>RAZ</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>WI, WRPERR</td>
</tr>
<tr>
<td></td>
<td>Erase</td>
<td></td>
</tr>
</tbody>
</table>
7.6.5 Flash memory register privileged and unprivileged modes

The flash registers can be read and written by privileged and unprivileged accesses depending on the SPRIV and NSPRIV bits in the FLASH  

privilege configuration register (FLASH_PRIVCFGR):

- When the SPRIV (respectively NSPRIV) bit is reset, all secure (respectively non-secure) flash registers can be read and written by both privileged or unprivileged access.
- When the SPRIV (respectively NSPRIV) bit is set, all secure (respectively non-secure) flash registers can be read and written by privileged access only. Unprivileged access to a privileged registers is RAZ/WI.

### Table 54. Privilege and mass or bank erase

<table>
<thead>
<tr>
<th>Access (TZ state not relevant)</th>
<th>Main flash</th>
<th>Unprivileged FLASH</th>
<th>Privileged FLASH</th>
<th>Mix unprivileged and privileged FLASH</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEP</td>
<td>OK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MENP</td>
<td>OK</td>
<td>WI, WRPERR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** For PRIVBBxy[i] access control, refer to Table 55 and Table 56.

### Table 55. Privilege configuration register access conditions (TZ enabled)

<table>
<thead>
<tr>
<th>Access</th>
<th>Corresponding sector secure status(1)</th>
<th>Sector setting access in PRIVBBxy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch Read</td>
<td>Secure/ non-secure</td>
<td>BUS ERROR</td>
</tr>
<tr>
<td>Write Privileged</td>
<td>Secure</td>
<td>OK (all sectors)</td>
</tr>
<tr>
<td>Write Non-secure</td>
<td>NS</td>
<td>OK (only the sector corresponding to the bit)</td>
</tr>
<tr>
<td>Write Unprivileged</td>
<td>Secure/ non-secure</td>
<td>WI (only the sector corresponding to the bit)</td>
</tr>
</tbody>
</table>

1. Either watermark or block-based.

### Table 56. Privilege configuration register access conditions (TZ disabled)

<table>
<thead>
<tr>
<th>Access</th>
<th>Sector setting access in PRIVBBRxy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch Read</td>
<td>BUS ERROR</td>
</tr>
<tr>
<td>Write Privileged</td>
<td>OK</td>
</tr>
<tr>
<td>Write Unprivileged</td>
<td>WI</td>
</tr>
</tbody>
</table>

---

276/3119  
RM0481 Rev 1
The registers related to key storage (FLASH_NSOBKCFGR, FLASH_SECOBKCFGR, FLASH_NSOBKKEYR, FLASH_SECOBKEYR) are exceptions to this rule and can be only accessed in privilege, regardless of other settings.

The next table summarizes flash registers access control.

### Table 57. Flash register accesses\(^{(1)}\)

<table>
<thead>
<tr>
<th>Access</th>
<th>Non-secure register</th>
<th>Secure register</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NSPRIV = 1</td>
<td>SPRIV = 1</td>
</tr>
<tr>
<td></td>
<td>NSPRIV = 0</td>
<td>SPRIV = 0</td>
</tr>
<tr>
<td>X</td>
<td>Secure/ non-secure</td>
<td>Privileged/ non-privileged</td>
</tr>
<tr>
<td>R/W</td>
<td>Secure</td>
<td>Privileged</td>
</tr>
<tr>
<td>R/W</td>
<td>Non-secure</td>
<td>Privileged</td>
</tr>
<tr>
<td>R/W</td>
<td>Unprivileged</td>
<td>RAZ, WI</td>
</tr>
</tbody>
</table>

1. Some registers are privilege only, not configurable.

### 7.6.6 Flash memory banks attributes in case of bank swap

The SWAP_BANK option bit modifies the address of each bank in the memory map. When SWAP_BANK is reset, flash Bank1 is mapped at the lower address range. When SWAP_BANK is set, flash Bank1 is mapped at the higher address range. flash bank attributes follow their bank contents so there is no need to modify their setting registers when swapping banks:

- Flash secure watermark x FLASH_SECWMx
- Flash write protection sector group FLASH_WRPSG (refer to Section 7.6.8)
- Hide protection in FLASH_HDPx registers
- Flash secure block based bank x register y FLASH_SECBBxy
- Flash privilege block based bank x register y FLASH_PRIVBBxy
- The data area configuration FLASH_EDATA

The SWAP_BANK is rendered immutable by setting BOOT_LOCK in the user OB. If TZEN is enabled, then SECBOOT_LOCK locks the option. If TZEN is disabled, then NSBOOT_LOCK locks the SWAP_BANK option.

Note: The BK_ECC bit in the FLASH ECC detection register (FLASH_ECCDETR) and FLASH ECC correction register (FLASH_ECCCORR), BKSEL bit in FLASH non-secure control register (FLASH_NSCR) and BKSEL bit in FLASH secure control register (FLASH_SECCR) always refers to Bank1 (respectively Bank2) when it is low (respectively high), regardless of the SWAP_BANK value.
Figure 30 shows how security attributes and protections behave in case of bank swap.

Figure 30. Protection attributes in case of bank swap illustration

7.6.7 Flash memory configuration protection

The memory uses hardware mechanisms to protect the following assets against unwanted or spurious modifications (such as software bugs):

- Option bytes change
- Write operations
- Erase commands
- Interrupt masking
The memory configuration registers protection is summarized in Table 58. Registers not present in this table are not protected by a key.

Table 58. Flash interface register protection summary

<table>
<thead>
<tr>
<th>Register name</th>
<th>Unlocking register</th>
<th>Protected asset</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLASH_NSCR</td>
<td>FLASH_NSKEYR</td>
<td>Write/erase control</td>
</tr>
<tr>
<td>FLASH_SECCR</td>
<td>FLASH_SECKEYR</td>
<td>Secure write/erase control</td>
</tr>
<tr>
<td>FLASH_OPTCR + all_PRG registers</td>
<td>FLASH_OPTKEYR</td>
<td>Flash bank option byte words change</td>
</tr>
<tr>
<td>FLASH_NSOBKCFGGR</td>
<td>FLASH_NSOBKKEYR</td>
<td>Flash OBTKey storage manipulation</td>
</tr>
<tr>
<td>FLASH_SECOBKCFGGR</td>
<td>FLASH_SECOBKKEYR</td>
<td>Flash OBTKey storage manipulation</td>
</tr>
</tbody>
</table>

7.6.8 Write protection

The purpose of the write protection is to prevent unwanted modifications to code and/or data.

Any flash group of four consecutive 8-Kbyte sectors can be independently write-protected or unprotected by clearing/setting the corresponding WRPSGn1/2 bit in the FLASH_WRP1R register (see FLASH write sector group protection for Bank 1 (FLASH_WRP1R_CUR) and FLASH write sector group protection for Bank 2 (FLASH_WRP2R_CUR)).

A write-protected group of sectors cannot be erased nor programmed. As a result, a bank erase cannot be performed if one group of sectors is write-protected, unless a NVSTATE transition to OPEN is triggered (erasing the whole user flash memory).

Note: Write protection errors are documented in Section 7.9.

7.6.9 Flash high-cycle data protections

The embedded flash memory can be configured to have 96-Kbyte memory area with high-cycling capability (100 kcycles) to store data and emulate EEPROM, see Section 7.3.10.

Option-byte controlled security features for user flash memory prevent sectors from being used as high-cycle data area. No OPTCHANGEERR is raised in case of setting conflict, it only makes the sectors unusable. Details are listed in Table 59.

Volatile settings using FLASH_SECBBxx and FLASH_PRIVBBxx can be applied. In this case the security setting of user flash sector 127 applies to flash high-cycle data sector 0 and so on, as indicated in Figure 23, Figure 24 and Figure 25.

Volatile setting of HDP extension that protrudes into sectors of data area also effectively causes the data area to be inaccessible.
### Table 59. High-cycle area protection summary: access to data area address range

<table>
<thead>
<tr>
<th>TZ protection (when TZ_STATE = 0xB4)(1)</th>
<th>Data sector enabled</th>
<th>Data sector not enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SecBB protected</td>
<td>NS sector</td>
</tr>
<tr>
<td></td>
<td>16/32 bits access</td>
<td>16/32 bits access</td>
</tr>
<tr>
<td>Secure</td>
<td>OB secure area</td>
<td>Other</td>
</tr>
<tr>
<td>Fetch</td>
<td>BUS ERROR</td>
<td>BUS ERROR</td>
</tr>
<tr>
<td>Read</td>
<td>OK</td>
<td>RAZ, ILAC</td>
</tr>
<tr>
<td>Write</td>
<td>If not WRP: OK else WI, WRPERR</td>
<td>WI, WRPERR, ILAFM</td>
</tr>
<tr>
<td>Erase</td>
<td>Same protection as in corresponding classic user flash sector</td>
<td>BUS ERROR</td>
</tr>
<tr>
<td>Non-secure</td>
<td>BUS ERROR</td>
<td>BUS ERROR</td>
</tr>
<tr>
<td>Fetch</td>
<td>RAZ, ILAC</td>
<td>OK</td>
</tr>
<tr>
<td>Read</td>
<td>WI, WRPERR, ILAFM</td>
<td>If not WRP: OK else WI, WRPERR</td>
</tr>
<tr>
<td>Write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Erase</td>
<td>Same protection as in corresponding classic user flash sector</td>
<td></td>
</tr>
</tbody>
</table>

1. In case of TZ_STATE = 0xC3, there is no secure state access, no SecBB and no Secure Area. For HDP and NS access the rules remain the same.

### Table 60. HDP protected definition

<table>
<thead>
<tr>
<th>User flash</th>
<th>OB HDP Area enabled</th>
<th>HDP extension enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDPL0 or HDPL1</td>
<td>HDP protected = 0</td>
<td>HDP protected = 0</td>
</tr>
<tr>
<td>HDPL2</td>
<td>HDP protected = 1</td>
<td>HDP protected = 0</td>
</tr>
<tr>
<td>HDPL3</td>
<td>HDP protected = 1</td>
<td>HDP protected = 1</td>
</tr>
</tbody>
</table>

### Table 61. Privilege sectors and the data area - access to data area address range

<table>
<thead>
<tr>
<th>Access</th>
<th>Data area address range access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unprivilege sector</td>
</tr>
<tr>
<td>Privilege</td>
<td>Fetch</td>
</tr>
<tr>
<td></td>
<td>Read</td>
</tr>
<tr>
<td></td>
<td>Write</td>
</tr>
<tr>
<td>Unprivilege</td>
<td>Fetch</td>
</tr>
<tr>
<td></td>
<td>Read</td>
</tr>
<tr>
<td></td>
<td>Write</td>
</tr>
</tbody>
</table>
7.6.10 Life cycle management

Non-volatile states, or debug states, are determined by the product state set by user option bytes. Debug possibility and possibility to change selected security settings are related to this state. While the state is stored in the OB, it is the SBS that controls the debug policy.

OBK storage access conditions are unaffected by the PRODUCT_STATE, but keys may be evicted in regression transitions.

HDP works equally regardless of the debug state. Even in debug the HDP area is hidden when HDPL increments.

The following states are defined:

<table>
<thead>
<tr>
<th>PRODUCT_STATE</th>
<th>Code in OB</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>0xED</td>
<td>User flash open (TZ secure and non-secure open). External access (debug) N and NS enabled (~RDP0).</td>
</tr>
<tr>
<td>Provisioning</td>
<td>0x17</td>
<td>Provisioning - immutable root of trust is being installed. Open if TZEN = 0xC3.</td>
</tr>
<tr>
<td>iROT-Provisioned</td>
<td>0x2E</td>
<td>The immutable root of trust is installed (~RDP 0.5).</td>
</tr>
<tr>
<td>TZ-Closed</td>
<td>0xC6</td>
<td>State for debugging the NS application. Debug is restrained to non-secure areas.</td>
</tr>
<tr>
<td>Closed</td>
<td>0x72</td>
<td>State for running a secure application. Debug disabled, regression is possible.</td>
</tr>
<tr>
<td>Locked</td>
<td>0x5C</td>
<td>Transition to other state, policy change or debug not permitted(^1).</td>
</tr>
<tr>
<td>Regression</td>
<td>0x9A</td>
<td>The temporary state initiated by the debug authentication system in transition to Open.</td>
</tr>
<tr>
<td>NS-Regression</td>
<td>0xA3</td>
<td>The temporary state initiated by the debug authentication system in transition to TZ-Closed.</td>
</tr>
</tbody>
</table>

1. The same situation is Closed state with incorrect or not defined debug certificates.

1. **No debug protection**

Read, program and erase operations into the flash main memory area are possible. Least restriction on the option bytes.

2. **Non-secure debug only**

All read and write operations (if no write protection is set) from/to the non-secure flash memory are possible. The debug access to secure area is prohibited. Debug access to non-secure area remains possible.
The following rules are enforced:

- **User mode**: code executing in user mode (boot flash) can access flash main memory and option bytes with all operations (read, erase, program).
- **Non-secure debug mode**: The secure flash memory areas are inaccessible; the non-secure flash memory areas remain accessible for debug purpose.
- **Boot RAM mode**: boot from SRAM is not possible.
- **Transition to Open state**: is possible with the consequence of flash mass erase and key slots revocation. Depends on debug unlock policy (see below).

### 3. Full product protection

All debug features are disabled and following rules enforced:

- **User mode**: code executing in user mode (boot flash) can access flash main memory and selected option bytes with all operations (read, erase, program).
- **Boot RAM mode**: boot from SRAM is not possible.
- **Transition to TZ-Closed or Open state**: is possible with the consequence of flash mass erase and key slots revocation. Depends on debug unlock policy (see below).

### 7.6.11 Product state transitions

Progressing to more closed state is the normal product life cycle, that does not require security measures. Transition in direction to open state is a regression, controlled by the debug authentication control. If debug unlock policy is set to “locked”, no regression is accepted. Locked state may be reinforced by invalidating the debug authentication certificates.

All transitions not listed in this chapter are invalid.

There is no restriction on reading the current product state.

#### Transitions in progress direction

Transition from Open to any of the closed states is matter of correct product configuration and provisioning. The transitions must be done in correct order.

The normal progression is:

- Open, Provisioning or iROT-Provisioned
- Provisioning to iROT-Provisioned
- iROT-Provisioned to Closed, Locked or Closed
- Closed to Locked or Closed

Transition is managed either by a software running on the device, or directly, using a debug interface. Transitions from Closed are only possible by software running on the device. By software it is assumed that transitions are triggered by bootloader, or root-of-trust services, but generally any software running on the device can do a progress transition.

#### Transition to Open state

This transition is a full regression. The starting state is any except Open and Locked. The debug tools are used to authenticate debug regression access rights with the debug authentication library, running on the device in HDPL1 (refer to Section 14.3.6: SBS debug control). After verifying the credentials, it puts the device into intermediate state Regression. From this state the device regresses securely in HDPL0 to Open.
The transition has the following consequence:
- Epoch secure counter is incremented
- Key slots revocation
- Product state updated
- User options updated (security settings reset, TZEN reset)
- Flash memory mass erase
- Backup RAM erase is requested

**Transition from closed to closed-secure debug state**

This transition opens the device to authorize the debug of the non-secure application software without compromising the security of the ROT functions.

The starting state is Closed. The debug tools are used to authenticate debug regression access rights with the debug authentication library, running on the device in the HDPL1 (refer to Section 14.3.6: SBS debug control). After verifying the credentials, it puts the device into intermediate state NS-Regression. From this state, the device regresses securely in HDPL0 to Closed.

The transition has the following consequence:
- Epoch non-secure counter is incremented
- Non-secure slots revocation
- Product state updated
- Flash memory sectors that are not covered by the secure watermark are erased
- Backup RAM erase is requested

### Table 63. PRODUCT_STATE transitions

<table>
<thead>
<tr>
<th>From</th>
<th>Open</th>
<th>Provisioning</th>
<th>iROT-Provisioned</th>
<th>TZ-Closed (2)</th>
<th>Closed</th>
<th>Locked</th>
<th>Regression</th>
<th>NS-Regression (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Provisioning</td>
<td>X</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OK (HDPL1)</td>
</tr>
<tr>
<td>iROT-Provisioned</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OK (HDPL1)</td>
</tr>
<tr>
<td>TZ-Closed</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OK (HDPL1)</td>
</tr>
<tr>
<td>Closed</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>OK</td>
<td>OK (HDPL1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OK (HDPL1)</td>
</tr>
<tr>
<td>Locked</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
In Table 63, X indicates transitions that are not permitted, dash indicates no change, OK indicates valid transitions. Some transitions are possible only in correct HDPL provided in parenthesis. Attempts to do an illegal transition result in OPTCHANGEERR.

Incorrect PRODUCT_STATE (no OBL) is interpreted as Locked.

7.6.12 OBK protection

At the receipt of an OBK access (read/write/execute), secure and privilege attributes are first checked, followed by OBK-HDPL. The OBK-HDPL value must exactly match the HDPL assigned to the key location, otherwise the OBKERROR flag is raised. Flash interface response is described in Table 64.

Table 63. PRODUCT_STATE transitions (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression (HDPL0)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>NS-Regression</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>OK (HDPL0)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

1. X = transitions not permitted (OPTCHANGEERR raised), - = no change, OK = valid transitions.
2. To transition into this state, both current and programmed value of TZEN must be enabled (TZEN = 0xB4), otherwise the change is ignored with OPTCHANGEERR.

Table 64. TZ OBK protection summary

<table>
<thead>
<tr>
<th>TZ protection (when TZ_STATE = 0xB4)</th>
<th>OBK access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No tamper</td>
</tr>
<tr>
<td>HDPL0/1/2/3</td>
<td>OBK selector</td>
</tr>
<tr>
<td>XS</td>
<td>BUS ERROR</td>
</tr>
<tr>
<td>RS</td>
<td>OK if privilege, otherwise RAZ</td>
</tr>
<tr>
<td>WS</td>
<td>OK if privilege, otherwise WI, WRPERR</td>
</tr>
<tr>
<td>XNS</td>
<td>BUS ERROR</td>
</tr>
<tr>
<td>RNS</td>
<td>RAZ, ILAFM</td>
</tr>
<tr>
<td>WNS</td>
<td>WI, WRPERR, ILAFM</td>
</tr>
<tr>
<td>Erase</td>
<td>NA</td>
</tr>
</tbody>
</table>

RAZ, WI
Note: Cacheable access is managed at system level and not in the flitf. The MPU needs to be configured to disable cache where it is not desirable.

In case of tamper, all keys are read as 0x00s.

There are also rules for accessing the OBKFGCR and OBKKEYR registers.

Table 65. OBK protection summary with TZ disabled

<table>
<thead>
<tr>
<th>TZ protection (when TZ_STATE = 0xC3)</th>
<th>OBK access</th>
<th>No tamper</th>
<th>Tamper</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDPL0/1/2/3</td>
<td>OBK selector</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XNS</td>
<td>BUS ERROR</td>
<td>BUS ERROR</td>
<td>RAZ, WI</td>
</tr>
<tr>
<td>RNS</td>
<td>OK if privilege, otherwise RAZ</td>
<td>RAZ</td>
<td></td>
</tr>
<tr>
<td>WNS</td>
<td>OK if privilege, otherwise WI, WRPERR</td>
<td>WI if privilege, otherwise WI, WRPERR</td>
<td></td>
</tr>
<tr>
<td>Erase</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

Table 66. Access conditions to secure control register

<table>
<thead>
<tr>
<th>S/NS</th>
<th>P/NP</th>
<th>TZ state</th>
<th>SECOBKCFGR and SECOBKKEYR access</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Bus error</td>
</tr>
<tr>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Bus error</td>
</tr>
<tr>
<td>R/W</td>
<td>S</td>
<td>P</td>
<td>Active</td>
</tr>
<tr>
<td>R/W</td>
<td>S</td>
<td>NP</td>
<td>Active</td>
</tr>
<tr>
<td>R/W</td>
<td>NS</td>
<td>P</td>
<td>Active</td>
</tr>
<tr>
<td>R/W</td>
<td>NS</td>
<td>NP</td>
<td>Active</td>
</tr>
<tr>
<td>R/W</td>
<td>-</td>
<td>-</td>
<td>Deactivated</td>
</tr>
</tbody>
</table>

Table 67. Access conditions to non-secure control register

<table>
<thead>
<tr>
<th>S/NS</th>
<th>P/NP</th>
<th>TZ state</th>
<th>NSOBKCFGR and NSOBKKEYR access</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Bus error</td>
</tr>
<tr>
<td>R/W</td>
<td>-</td>
<td>-</td>
<td>Active</td>
</tr>
<tr>
<td>R/W</td>
<td>-</td>
<td>P</td>
<td>Deactivated</td>
</tr>
<tr>
<td>R/W</td>
<td>-</td>
<td>NP</td>
<td>Deactivated</td>
</tr>
</tbody>
</table>

7.6.13 One-time-programmable and read-only memory protections

Sections of OTP/RO in the flash memory are described in details in Section 7.3.9. There is no protection provided by the flash memory interface other than the dedicated write protection.
No write is possible to the RO area, once it was established in manufacturing. The OTP write protection describes a method of locking out the OTP area.

The access conditions are summarized in **Table 68**.

<table>
<thead>
<tr>
<th>XS</th>
<th>BUS ERROR</th>
<th>BUS ERROR</th>
<th>BUS ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS</td>
<td>RAZ, ILAC</td>
<td>RAZ, ILAC</td>
<td>RAZ, ILAC</td>
</tr>
<tr>
<td>WS</td>
<td>WI, WRPERR, ILAFM</td>
<td>WI, WRPERR, ILAFM</td>
<td>WI, WRPERR, ILAFM</td>
</tr>
<tr>
<td>XNS</td>
<td>BUS ERROR</td>
<td>BUS ERROR</td>
<td>BUS ERROR</td>
</tr>
<tr>
<td>RNS</td>
<td>BUS ERROR</td>
<td>BUS ERROR</td>
<td>BUS ERROR</td>
</tr>
<tr>
<td>WNS</td>
<td>BUS ERROR</td>
<td>BUS ERROR</td>
<td>BUS ERROR</td>
</tr>
</tbody>
</table>

1. Word size is 16 bits. 32-bit access is possible. Other access attempt leads to bus error.

### 7.7 System memory

#### 7.7.1 Introduction

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

When TrustZone is enabled (user sets TZEN bitfield to 0xB4) then RSS provides secure services to secure user firmware only; when TrustZone is disabled (user sets TZEN bitfield to 0xC3), RSS provides services to user firmware.

#### 7.7.2 RSS user functions

The RSS provides runtime services thanks to RSS library, whose functions are exposed to user within the CMSIS device header file provided by the STM32CubeH5 firmware package (see UM3065 "Getting started with STM32CubeH5 for STM32H5 Series" for more details).

RSS provides services through two different libraries, RSSLIB and NSSLIB, dedicated to secure user firmware when system is configured, respectively, with TrustZone enabled (TZEN bitfield set to 0xB4) and TrustZone disabled (TZEN bitfield set to 0xC3).

#### Table 69. Macros for RSS services

<table>
<thead>
<tr>
<th>TrustZone</th>
<th>C defined macro</th>
<th>Location in flash memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>RSSLIB_PFUNC</td>
<td>0xBF9FB78</td>
</tr>
<tr>
<td>Disabled</td>
<td>NSSLIB_PFUNC</td>
<td>0xBF9FB6C</td>
</tr>
</tbody>
</table>
RSSLIB

The secure user firmware calls RSSLIB functions using RSSLIB_PFUNC C defined macro, which points to a location within non-secure system memory. Before calling RSSLIB functions, the secure user firmware must define a non-secure region above this location within SAU of the Cortex®-M33, starting from RSSLIB_SYS_FLASH_NS_PFUNC_START (0xBF9FB78), up to RSSLIB_SYS_FLASH_NS_PFUNC_END (0xBF9FB84). These last addresses are provided within the CMSIS device header file.

The user can set this non-secure region either by using the CMSIS system partition header file, or by implementing its own code for SAU setup. The CMSIS system partition header file is part of the STM32CubeH5 firmware package.

<table>
<thead>
<tr>
<th>Library</th>
<th>Function</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSSLIB_PFUNC</td>
<td>JumpHDPLvl2</td>
<td>Secure callable function</td>
</tr>
<tr>
<td></td>
<td>JumpHDPLvl3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JumpHDPLvl3NS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RSSLIB_DataProvisioning</td>
<td>Non-secure callable function</td>
</tr>
</tbody>
</table>

RSSLIB_DataProvisioning

Secure attribute: Non-secure callable function.

Prototype:
```
uint32_t RSSLIB_DataProvisioning_(RSSLIB_DataProvisioningConf_t *pConfig)
```

Arguments:

pConfig: input parameter. RSSLIB_DataProvisioningConf_t

C structure definition is described below:

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

Structure elements

- **pSource**: Provides the address of data to be provisioned. Must be within SRAM3 address range (non-secure aliases).
- **pDestination**: Provides the address where to store data to be provisioned. Must be within OBKeys address range. To be aligned on 16 bytes.
- **Size**: Provides the size of data to be provisioned (the number of bytes, must be a multiple of 16).
DoEncryption
Notifies RSSLIB_DataProvisioning function if it must encrypt or not data within OBKeys.

- DoEncrypt can be either 0xF5F50AAU or 0xCACA0AA0U (the only one allowed on STM32H563xx devices)
- 0xF5F50AA0U: notifies RSSLIB_DataProvisioning to encrypt data with relevant DHUK before programming it within OBKeys. DHUK is selected according to pDestination value.
- 0xCACA0AA0U: notifies RSSLIB_DataProvisioning to program in clear data within OBKeys.

Crc
CRC over full source data buffer, pConfig->pDestination value, pConfig->Size value and finally pConfig->DoEncryption value.

CRC computation uses CRC-32 (Ethernet):
- CRC polynomial: 0x04C11DB7U
- Initial value: 0xFFFFFFFFU

Returned values
0xEAEAEAEAU Success
0xF5F5E0E0U Error:
  - pConfig or pSource are not in SRAM3
  - pDestination is not aligned on 16 bytes
  - pDestination + Size does not fit within an OBKey section

0xF5F550E0EU Error: Size is not a multiple of 16
0xF5F55080U Error: computed CRC is not the expected one provided within pConfig->CRC
0xF5F55008U Error: cannot program data within OBKeys destination section
0xF5F55E00EU Error: wrong DoEncryption parameter value
0xF5F55080U Error: encryption requested, butw platform does not support it
0xF5F550E0EU Error: encryption error
0xF5F55080U. Error: OBKeys programming error

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

The destination address (pDestination) added to the size (Size) of the data to be provisioned must not cross the OBKeys section boundaries, listed below.

<table>
<thead>
<tr>
<th>OBKeys Level1</th>
<th>Start address: 0xFFFFD0100UL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>End address: 0xFFFFD08FFUL</td>
</tr>
<tr>
<td>OBKeys Level2</td>
<td>Start address: 0xFFFFD0900UL</td>
</tr>
<tr>
<td></td>
<td>End address: 0xFFFFD0BFFUL</td>
</tr>
<tr>
<td>OBKeys Level3 secure</td>
<td>Start address: 0xFFFFD0C00UL</td>
</tr>
<tr>
<td></td>
<td>End address: 0xFFFFD17FFUL</td>
</tr>
<tr>
<td>OBKeys Level3 non-secure</td>
<td>Start address: 0xFFFFD1800UL</td>
</tr>
<tr>
<td></td>
<td>End address: 0xFFFFD1FEFUL</td>
</tr>
</tbody>
</table>
When requested through `pConfig->DoEncryption` parameter, RSSLIB_DataProvisioning encrypts data before programming them within OBKeys.

RSSLIB_DataProvisioning uses AES CBC 128 bits with:

- IV: defined as (using C definition format)
  
  ```c
  uint32_t IV = {0x8001D1CEU, 0xD1CED1CEU, 0xD1CE8001U, 0xCED1CED1U};
  ```

- Key: DHUK corresponding to the targeted OBKeys section defined by `pConfig->pDestination` parameter.

**Note:** Data encryption within OBkeys is supported only by STM32H573xx devices.

**JumpHDPLvl2**

Secure attribute: Secure callable function.

Prototype:

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

User code function call example:

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

Arguments:

- **VectorTableAddr:**
  
  - Input parameter, address of the next vector table to apply.
  
  - The vector table format is the one used by the Cortex-M33 core.

- **MPUIndex:**
  
  - Input parameter, MPU region index. Caller function must define (but keep disabled) the corresponding MPU region before calling JumpHDPLvl2. The function enables the MPU region before jumping to the reset handler of the vector table. The vector table reset handler function belongs to the MPU region.

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

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

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

In case of failure (bad input parameter value), RSSLIB_Sec_JumpHDPLvl2 returns 0xF5F5F5F5.

**JumpHDPLvl3**

Secure attribute: Secure callable function.
Prototype:

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

User code function call example:
```
RSSLIB_PFUNC->S.JumpHDPLvl3((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:
- **VectorTableAddr:**
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex-M33 core.
- **MPUIndex:**
  - Input parameter, MPU region index. Caller function must define but keep disabled the corresponding MPU region before calling JumpHDPLvl3. The function enables the MPU region before jumping to the reset handler of the vector table, whose function belongs to the MPU region.

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

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

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

**JumpHDPLvl3NS**

Secure attribute: Secure callable function.

Prototype:

```c
uint32_t JumpHDPLvl3NS(uint32_t VectorTableAddr)
```

User code function call example:
```
RSSLIB_PFUNC->S.JumpHDPLvl3NS((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:
- **VectorTableAddr:**
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex-M33 core.

User calls JumpHDPLvl3NS to close user Flash HDPL1 and HDPL2 areas by incrementing HDPL up to 3, to move from secure to non-secure domain, then jump to the non-secure...
reset handler embedded within the vector table, whose address is passed as input parameter.

After closing HDPL1/2, JumpHDPLvl3 jumps to the non-secure reset handler function supported by the vector table. JumpHDPLvl3NS does not set the new vector table.

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

In case of failure (bad input parameter value), JumpHDPLvl3NS returns 0xF5F5F5F5.

**NSSLIB**

The user firmware calls only the NSSLIB function when TrustZone is disabled (TZEN bitfield is set to 0xC3).

The user firmware calls NSSLIB functions using NSSLIB_PFUNC C defined macro, which points to a location within system memory.

### Table 71. NSS lib interface functions

<table>
<thead>
<tr>
<th>Library</th>
<th>Function</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSSLIB_PFUNC</td>
<td>JumpHDPLvl2</td>
<td>Non-secure function</td>
</tr>
<tr>
<td></td>
<td>JumpHDPLvl3</td>
<td></td>
</tr>
</tbody>
</table>

**JumpHDPLvl2**

Prototype:

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

User code function call example:

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

Arguments:

- **VectorTableAddr:**
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex-M33 core.
- **MPUIndex:**
  - Input parameter, MPU region index. Caller function shall define but keep disable the corresponding MPU region before calling JumpHDPLvl2. The function enables the MPU region before jumping to the reset handler of the vector table. The vector table reset handler function belongs to the MPU region.

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

After closing HDPL1, JumpHDPLvl2 enables the MPU region provided as input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed vector table and jumps to the reset handler function supported by the vector table too. JumpHDPLvl2 does not set the new vector table.
On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), RSSLIB_Sec_JumpHDPLvl2 returns 0xF5F5F5F5.

**JumpHDPLvl3**

Prototype:

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

User code function call example:

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

Arguments:

- **VectorTableAddr**:
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex-M33 core.
- **MPUIndex**:
  - Input parameter, MPU region index. Caller function shall define but keep disable the corresponding MPU region before calling JumpHDPLvl3. The function enables the MPU region before jumping to the reset handler of the vector table. The vector table reset handler function belongs to the MPU region.

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

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

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

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

### 7.8 FLASH low-power modes

Table 72 summarizes the memory behavior in STM32 low-power modes. Embedded flash memory belongs to the core domain.

<table>
<thead>
<tr>
<th>Power mode</th>
<th>Core domain voltage range</th>
<th>Allowed if FLASH busy</th>
<th>FLASH power mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>VOS0/1/2/3</td>
<td>Yes</td>
<td>Run</td>
</tr>
<tr>
<td>Stop1 (clock stopped)</td>
<td>SVOS3/4/5</td>
<td>No</td>
<td>Clock gated or stopped in case of SVOS5</td>
</tr>
<tr>
<td>Standby</td>
<td>Off</td>
<td>No</td>
<td>Off</td>
</tr>
</tbody>
</table>
When the system state changes or within a given system state, the memory can operate with a different voltage supply range (VOS), according to the application. The procedure to switch the memory into power modes (run, clock gated, stopped, off) is described hereafter.

Note: For more information on the microcontroller power states, refer to Section 10: Power control (PWR).

Managing the FLASH domain switching to Stop or Standby

As explained in Table 72, if the memory informs the reset and clock controller (RCC) that it is busy (BSY, DBNE, WBNE is set), the microcontroller cannot switch the core domain to Stop or Standby mode.

There are two ways to release the memory:

- Reset the WBNE busy flag in FLASH_NS/SECSR register by any of the following actions:
  - Complete the write buffer with missing data.
  - Force the write operation without filling the missing data by activating the FW bit in FLASH_NS/SECCR register. This forces all missing data “high”.
- Poll BSY busy bits in FLASH_NS/SECSR register until they are cleared. This indicates that all recorded write, erase and option change operations are complete.

The microcontroller can then switch the domain to Stop or Standby mode.

7.9 FLASH error management

7.9.1 Introduction

The memory automatically reports when an error occurs during a read, program or erase operation. A wide range of errors are reported:

- Non-secure write protection error (WRPERR)
- Secure write protection error (WRPERR)
- Non secure programming sequence error (PGSERR)
- Secure programming sequence error (PGSERR)
- Secure strobe error (STRBERR)
- Error correction code error (ECCC, ECCD)
- Illegal access (ILAFM/ILAP)
- Option-byte change error (OPTCHANGEERR)
- OBK non-secure general error (OBKERR)
- OBK secure general error (OBKERR)
- OBK non-secure write error (OBKWERR)
- OBK secure write error (OBKWERR)

The application software can individually enable the interrupt for each error, as detailed in Section 7.10.

The flash memory interface uses different interrupt lines to trigger the event. There are two direct lines to NVIC, one for secure interrupts, other for non secure. Third line leads to TZIC, for the ILAFM interrupt.
Note: For all errors, the application software must clear the error flag before attempting a new modify operation.

Since there is just one write buffer and only one operation is allowed at a time, the write control bits and status bits are shared by both banks:

- Write control bits (PG, FW, EOPIE, WRPERRIE, PGSERRIE, STRBERRIE, INCERRIE, CLR_EOP, CLR_WRPERR, CLR_STRBERR, CLR_INCERR, CLR_PGSERR) controls bank 1 and 2 at the same time.
- Write status flag reports (WBNE, DBNE, EOP, WRPERR, PGSERR, STRBERR, INCERR, BSY) error for Bank 1 and 2 at the same time.

7.9.2 Non-secure write protection error (WRPERR)

When an illegal non-secure erase/program operation is attempted to the nonvolatile memory, the flash interface sets the write protection error flag WRPERR in FLASH_NSSR register.

An erase operation is rejected and flagged as illegal if it targets one of the following memory areas:

- A sector write-protected with WRPSGn
- An HDP area while the HDPL has made it inaccessible
- Secure (TZ) state not matching the erased memory attribute
- Attempt to erase privilege sector within unprivileged mode

A program operation is ignored and flagged as illegal if it targets one of the following memory areas:

- System flash memory
- A user sector write-protected with WRPSGn
- An OTP block, locked with LOCKBL
- A read-only section
- A reserved area
- Secure area
- Privileged area from within unprivileged mode
- OBK access conditions check error

When WRPERR flag is raised, the operation is rejected and nothing is changed in the corresponding bank. If this error is detected, the write buffer is invalidated.

Note: WRPERR flag must be cleared before any erase/program operation.

WRPERR flag is cleared by setting CLR_WRPERR bit to 1 in FLASH_NSCCR register.

If WRPERRIE bit in FLASH_NSCTR register is set to 1, an interrupt is generated when WRPERR flag is raised (see Section 7.10 for details).

7.9.3 Secure write protection error (WRPERR)

When an illegal secure erase/program operation is attempted to the nonvolatile memory, the flash interface sets the write protection error flag WRPERR in FLASH_SECSR register.
An erase operation is rejected and flagged as illegal if it targets one of the following memory areas:
- A sector write-protected with WRPSGn
- An HDP area while the HDPL has made it inaccessible
- Secure (TZ) state not matching the erased memory attribute
- Attempt to erase privilege sector within unprivileged mode

A program operation is ignored and flagged as illegal if it targets one of the following memory areas:
- System flash memory
- A user sector write-protected with WRPSGn
- An OTP block, locked with LOCKBL
- A read-only section
- A reserved area
- Non-secure area
- Privileged area from within unprivileged mode
- OBK access condition check error

When WRPERR flag is raised, the operation is rejected and nothing is changed in the corresponding bank.

If this error is detected the write buffer is invalidated.

**Note:** WRPERR flag must be cleared before any erase/program operation.

WRPERR flag is cleared by setting CLR_WRPERR bit to 1 in FLASH_SECCCR register.

If WRPERRIE bit in FLASH_SECCR register is set to 1, an interrupt is generated when WRPERR flag is raised (see Section 7.10 for details).

### 7.9.4 Non secure programming sequence error (PGSERR)

When the non-secure programming sequence is incorrect, the flash interface sets the programming sequence error flag PGSERR in FLASH_NSSR register.

More specifically, PGSERR flag is set when one of below conditions is met:
- An error like INCERR, WRPERR, PGSERR, STRBERR, OPTCHANGEERR, OBKERR or OBKWERR have not been cleared before requesting a new write or erase operation, OBK operation or options change.
- For erase, PGSERR is set if:
  - missing erase operation (FLASH_NSCR): STRT = 1 with (MER = 0, SER = 0 and BER = 0)
  - sector and bank erase requested at the same time (NSSTRT and more than one of MER, SER and BER)
  - PG set during erase operation (it ensures that no erase req and write req happen at the same time): (STRT = 1) with (PG = 1)
  - Erase operation is started while write buffer is waiting for next data: (STRT = 1) with WBNE = 1.
  - STRT = 1 set by secure access.
  - Erase operation is started while DBNE = 1.
For programming, PGSERR is set if:
- missing write flag: A write operation is requested but the program enable bit PG has not been set in FLASH_NSCR register prior to the request.
- write operation to flash high-cycle data is requested but PG = 0.
- AHB write request is received and (SER = 1, BER = 1 or MER = 1)
- 16 bit data access requested while WBNE = 1

For options change, PGSERR is set if:
- option change is started while write buffer is waiting for next data: OPTSTRT = 1 with WBNE = 1
- OPTSTRT set with non-secure DBNE = 1.

For options byte key storage access, PGSERR is set if:
- error flags from previous operation not cleared
- ALT_SECT_ERASE and SWAP_SECT_REQ in FLASH_NSOBKCFGR are set at the same time
- SWAP_OFFSET value is wrong: when SWAP_SECT_REQ is set to 1, the SWAP_OFFSET should be equal or greater than OBK_HDPL< N - 1 > _SWAP_OFFSET
- ALT_SECT_ERASE or OBK_SWAP set while DBNE = 1
- OBK-HDPL is incorrect and non-secure OBK_SWAP is requested
- If OBKSWAP or OBK alternate erase is started while write buffer is waiting for new data (WBNE = 1).

When PGSERR flag is raised as consequence of failed write attempt (flash programming), the current program operation is aborted and nothing is changed in the corresponding bank. The write data buffer is also invalidated.

**Note:** When PGSERR flag is raised, there is a risk that the last write operation performed by the application has been lost because of the above protection mechanism. It is recommended to generate interrupts on PGSERR and verify in the interrupt handler if the last write operation has been successful, by reading back the value in the flash memory.

The PGSERR flag also blocks any new program operation. This means that PGSERR must be cleared before starting a new program operation.

PGSERR flag is cleared by setting CLR_PGSERR bit to 1 in FLASH_NSCCR register.

If PGSERRIE bit in FLASH_NSCR register is set to 1, an interrupt is generated when PGSERR flag is raised. See Section 7.10 for details.
Secure programming sequence error (PGSERR)

When the secure programming sequence is incorrect, the flash interface sets the programming sequence error flag PGSERR in FLASH_SECSR register.

More specifically, PGSERR flag is set when one of below conditions is met:

- An error like INCERR, WRPERR, PGSERR, STRBERR, OBKERR, or OBKWERR has not been cleared before requesting a new write or erase operation, OBK operation or options change.

- For erase, PGSERR is set if:
  - missing erase operation (FLASH_SECCR): STRT = 1 with (SER = 0 and BER = 0, MER = 0)
  - sector and bank erase requested at the same time (SECSTRT and more than one of MER, SER, and BER)
  - PG set during erase operation (it ensures that no erase request and write request happen at the same time): (STRT = 1) with (PG = 1)
  - erase operation is started while write buffer is waiting for next data: (STRT = 1) with WBNE = 1.
  - operation started while secure DBNE = 1.

- For programming, PGSERR is set if:
  - missing write flag: a write operation is requested but the program enable bit PG has not been set in FLASH_SECCR register prior to the request
  - write operation to flash high-cycle data is requested but PG = 0
  - AHB write request is received and (SER = 1, SBER = 1, or MER = 1)
  - 16 bit data access requested while WBNE = 1

- For options change, PGSERR is set if:
  - Option change is started while write buffer is waiting for next data: OPTSTRT = 1 with WBNE = 1
  - OPTSTRT set with secure DBNE = 1

- For options byte key storage access, SECPGSERR is set if:
  - error flags from previous operation not cleared
  - ALT_SECT_ERASE and SWAP_SECT_REQ are set at the same time
  - SWAP_OFFSET value is wrong: when SWAP_SECT_REQ = 1, SWAP_OFFSET must be equal or greater than OBK_HDPL<N-1>_SWAP_OFFSET.
  - SWAP_SECT_REQ or OBK_SWAP set while DBNE = 1
  - OBK-HDPL is incorrect and secure OBK_SWAP is requested
  - If OBKSWAP or OBK alternate erase is started while write buffer is waiting for new data (WBNE = 1)

When PGSERR flag is raised as consequence of failed write attempt (flash programming), the current program operation is aborted and nothing is changed in the corresponding bank. The write data buffer is also invalidated.

Note: When PGSERR flag is raised, there is a risk that the last write operation performed by the application has been lost because of the above protection mechanism. It is recommended to generate interrupts on PGSERR and verify in the interrupt handler if the last write operation has been successful by reading back the value in the flash memory.
The PGSERR flag also blocks any new program operation. This means that PGSERR must be cleared before starting a new program operation.

PGSERR flag is cleared by setting CLR_SECPGSERR bit to 1 in FLASH_SECCCR register.

If PGSERRIE bit in FLASH_SECCR register is set to 1, an interrupt is generated when PGSERR flag is raised. See Section 7.10 for details.

7.9.6 Non-secure strobe error (STRBERR)

When the non-secure application software writes several times to the same byte in the write buffer, the flash interface sets the strobe error flag STRBERR (FLASH_NSSR) whatever the target bank of the write access.

When STRBERR flag is raised, the current program operation is aborted and the write buffer is invalidated.

STRBERR flag is cleared by setting CLR_STRBERR bit to 1 in FLASH_NSCCR register.

If STRBERRIE bit in FLASH_NSCR register is set to 1, an interrupt is generated when STRBERR flag is raised. See Section 7.10 for details.

7.9.7 Secure strobe error (STRBERR)

When the secure application software writes several times to the same byte in the write buffer, the memory sets the strobe error flag STRBERR (FLASH_SECSR) whatever the target bank of the write access.

When STRBERR flag is raised, the current program operation is aborted and the write buffer is invalidated.

STRBERR flag is cleared by setting CLR_STRBERR bit to 1 in FLASH_SECCCR register.

If STRBERRIE bit in FLASH_SECCR register is set to 1, an interrupt is generated when STRBERR flag is raised. See Section 7.10 for details.

7.9.8 Non-secure inconsistency error (INCERR)

When a programming inconsistency in non-secure access is detected, the flash interface sets the inconsistency error flag INCERR in register FLASH_NSSR.

More specifically, INCERR flag is set when one of the following conditions is met:

- A write operation is attempted before completion of the previous write operation, for example:
  - The application software starts a write operation to fill the 128-bit write buffer, but sends a new write burst request to a different flash memory address before the buffer is full.
  - One master starts a write operation, but before the buffer is full, another master starts a new write operation to the same address or to a different address.
  - ALT_SECT in FLASH_NSOBKCFGR changed while filling the write buffer.

Note: **INCERR flag must be cleared before starting a new write operation, otherwise a sequence error (PGSERR) is raised.**

It is recommended to follow the following sequence to avoid losing data when an inconsistency error occurs:
1. Execute a handler routine when INCERR flag is raised.
2. Stop all write requests to the memory.
4. Restart the write operations from where they have been interrupted.

INCERR flag is cleared by setting CLR_INCERR bit to 1 in FLASH_NSCCR register.

If INCERRIE bit in FLASH_NSCR register is set to 1, an interrupt is generated when INCERR flag is raised (see Section 7.10 for details).

7.9.9 Secure inconsistency error (INCERR)

When a programming inconsistency is detected, the flash interface sets the inconsistency error flag INCERR in register FLASH_SECSR.

More specifically, INCERR flag is set when one of the following conditions is met:
- A write operation is attempted before completion of the previous write operation, for example:
  - The application software starts a write operation to fill the 128-bit write buffer, but sends a new write burst request to a different flash memory address before the buffer is full.
  - One master starts a write operation, but before the buffer is full, another master starts a new write operation to the same address or to a different address.

Note: INCERR flag must be cleared before starting a new write operation, otherwise a sequence error (PGSERR) is raised.

It is recommended to follow the sequence below to avoid losing data when an inconsistency error occurs:
1. Execute a handler routine when INCERR flag is raised.
2. Stop all write requests to the flash memory.
4. Restart the write operations from where they have been interrupted.

INCERR flag is cleared by setting CLR_INCERR bit to 1 in FLASH_SECCCR register.

If INCERRIE bit in FLASH_SECCR register is set to 1, an interrupt is generated when INCERR flag is raised (see Section 7.10 for details).

7.9.10 Error correction code error (ECCC, ECCD)

When a single-error correction is detected during a read, the flash interface sets the single-error correction flag ECCC in FLASH_ECCCORR register.

When two ECC errors are detected during a read, the flash interface sets the double error detection flag ECCD in FLASH_ECCDETR register.

When the ECCC flag is raised, the corrected read data are returned. The application can ignore the error, and request new read operations. When the ECCD the flag is raised, an NMI is generated, it can be masked in SBS registers (SBS flift ECC NMI mask register (SBS_ECCNMIR)) for data access (OTP, data area, RO data). Software must invalidate the instruction cache (CACHEINV = 1) in the NMI interrupt service routine when the ECCD flag is set.
When ECCC or ECCD flag is raised, the address of the flash word that generated the error is saved in the FLASH_ECCCORR (FLASH_ECCDETR) register. If the address corresponds to a read-only area or to an OTP area or flash high-cycle data, the OTP_ECC bit is also set to 1 in the FLASH_ECCCORR (FLASH_ECCDETR) register. This register is automatically cleared when the associated flag that generated the error is reset.

A BK_ECC flag indicates in which flash bank the error occurred.

A SYSF_ECC flag indicates an error detected in the system flash area.

### Table 73. Locating ECC failure

<table>
<thead>
<tr>
<th>OTP_ECC</th>
<th>SYSF_ECC</th>
<th>BK_ECC</th>
<th>EDATA_ECC</th>
<th>OBK_ECC</th>
<th>Flash area</th>
<th>ADDR_ECC min</th>
<th>ADDR_ECC max</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0/1</td>
<td>0</td>
<td>0</td>
<td>User flash</td>
<td>0x0000</td>
<td>0xFFFF</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0/1</td>
<td>0</td>
<td>0</td>
<td>System flash</td>
<td>0x0000</td>
<td>0x0FFF</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>OTP</td>
<td>0x0600</td>
<td>0x07FF</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>OBKeys</td>
<td>0x0000</td>
<td>0x03FF</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0/1</td>
<td>1</td>
<td>0</td>
<td>Data area sector 7</td>
<td>0xF000</td>
<td>0xF1FF</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0/1</td>
<td>1</td>
<td>0</td>
<td>Data area sector 6</td>
<td>0xF200</td>
<td>0xF3FF</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0/1</td>
<td>1</td>
<td>0</td>
<td>Data area sector 5</td>
<td>0xF400</td>
<td>0xF5FF</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0/1</td>
<td>1</td>
<td>0</td>
<td>Data area sector 4</td>
<td>0xF600</td>
<td>0xF7FF</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0/1</td>
<td>1</td>
<td>0</td>
<td>Data area sector 3</td>
<td>0xF800</td>
<td>0xF9FF</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0/1</td>
<td>1</td>
<td>0</td>
<td>Data area sector 2</td>
<td>0xFA00</td>
<td>0xFBFF</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0/1</td>
<td>1</td>
<td>0</td>
<td>Data area sector 1</td>
<td>0xFC00</td>
<td>0xFDFF</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0/1</td>
<td>1</td>
<td>0</td>
<td>Data area sector 0</td>
<td>0xFE00</td>
<td>0xFFFF</td>
</tr>
</tbody>
</table>

**Note:** In case of successive single correction or double detection errors, only the address corresponding to the first error is stored in FLASH_ECCCORR (FLASH_ECCDETR) register.

It is mandatory to clear ECCC or ECCD flags before starting a new read operation.

As the ECC interface is shared by the two banks, if the same error is registered simultaneously, a physical Bank1 error is registered. Both errors are registered if one bank reports ECCD and the other bank ECCC.

ECCC (respectively ECCD) flag is cleared by setting to 1 ECCC bit (respectively ECCD bit) in FLASH_ECCCORR (FLASH_ECCDETR) register.

If ECCC bit in FLASH_ECCCORR register is set to 1, an interrupt is generated when ECC flag is raised. Only NMI is generated for the ECCD. See Section 7.10 for details.
7.9.11 Illegal access (ILAFM/ILAP)

Illegal access is a signal to the TZIC, triggering a secure interrupt in there. It is complementary to the other interrupts and only generated when the TZ is enabled (TZ_STATE = 0xB4).

It can be masked only on GTZC level.

ILAFM is generated when rules on secure flash memory access are violated:
- Attempt to access secure memory location in non-secure mode.
- Attempt to access non-secure memory location in secure mode.

ILAP is generated on:
- Attempt to access secure register in non-secure mode.

If ILAFM is detected during write, the write buffer is invalidated.

Note: ILAFM and ILAP has no flag within the flash memory controller to clear to resume operation. It is dealt within the TZIC.

7.9.12 Option-byte change error (OPTCHANGEERR)

When the flash interface finds an error during an option change operation, it aborts the operation and sets the option byte change error flag OPTCHANGEERR in FLASH_NSSR register.

The error is raised after OPTSTRT bit set if:
- TZEN is not set and transition to TZ-Closed or NS-Regression state is requested. Also any other forbidden PRODUCT_STATE transition (Table 63).
- SWAP_BANK is locked out by combination of TZEN and BOOT_LOCK.
- OB modification is attempted in wrong HDPL (selected PRODUCT_STATE transitions)
- Attempt to modify OB with invalid value
  - New EPOCH value not greater than current
  - Value not recognized by the specification (enumerated magic numbers only)
- Attempt to modify OB in product state where it is not allowed (usually a state open for debug is required for change).

Note: Exceptions and details provided in the OB description (see Section 7.4.7 and Table 43).

OPTCHANGEERR flag is cleared by setting CLR_OPTCHANGEERR bit to 1 in FLASH_NSCCR register.

If OPTCHANGEERRIE bit in FLASH_NSCR register is set to 1, an interrupt is generated when OPTCHANGEERR flag is raised (see Section 7.10 for details).

It is mandatory to clean the OPTCHANGEERR flag before starting a new operation (option change, erase or write).
7.9.13 **Miscellaneous HardFault errors**

The following events generate a bus error on the corresponding bus interface:
- On main AHB system bus for access targeting code and data with 9 bits ECC:
  - Fetching from secure user flash memory in non-secure mode.
  - Fetching from non-secure user flash memory in secure mode.
  - Access to invalid address (including data addresses forbidden for code use).
  - Fetching from HDP area with incorrect HDPL value.
  - Fetching from privilege area in unprivileged mode.
- On AHB configuration or system bus for accesses targeting OTP/RO (all addresses using 6 bits ECC):
  - Wrong key input to FLASH_NS/SECKEYR or FLASH_OPTKEYR.
  - 8-bit accesses to system AHB interface.
  - Wrong unlock sequence on a register.

7.9.14 **OBK error cases (OBKERR, OBKWERR)**

Four error types can be detected while filling the OBK write buffer: STRBERR, INCERR, WRPERR or OBKERR. The OBKERR is specific to the OBK access and cannot be raised by access to any other flash memory asset.

*Note:* For STRBERR, INCERR and WRPERR, non-secure or secure flags are used depending on the AHB access type (sec or non-secure). If a non-secure access is received, the non-secure error flags are used. If a secure access is received, the secure error flags are used.

For OBKERR and OBKWERR, non-secure or secure flags are used depending on the value of the secure state. If the SOC is in secure state (TZ_STATE active), the secure error flags (OBKERR and OBKWERR) are used. Otherwise the non-secure error flags (OBKERR and OBKWERR) are used.

**OBKERR** is raised when:
- the OBK-HDPL (input signal from SBS) does not match the HDPL associated to the key during a read or write access, and the TZ and PRIV attributes are correct (See Section 7.5.2 for more details).

If OBKERR is raised during write access, the write buffer is flushed. If the OBKERR is raised on read access, the write buffer is intact. In both cases the flag must be cleared to enable write/erase access to flash memory.

**OBKWERR** is detected after filling the write buffer and raised if:
- the address is not virgin during a write access.
- OBK selector in the alternate sector is not virgin during a SWAP operation.

If OBKWERR is raised, the write buffer is flushed. The flag must be cleared to enable write/erase access to flash memory.
7.10 FLASH interrupts

The flash interface can generate a maskable interrupt to signal the following events on a given bank:

- Read and write errors (see Section 7.9)
  - Single ECC error correction during read operation
  - Write inconsistency error
  - Bad programming sequence
  - Strobe error during write operations
  - option change operation error
- Security errors (see Section 7.9)
  - Write protection error
  - Illegal access error - signal to TZIC
- Miscellaneous events (described below)
  - End of programming

A NMI is raised on double ECC error detection during read operation.

Two interrupt lines gather all the error sources, one for errors in non-secure access and second for errors that happen in secure access.

The user can individually enable or disable flash interface interrupt sources by changing the mask bits in the FLASH_NS/SECCR, FLASH_ECCCORR and FLASH_ECCDETR register. Setting the appropriate mask bit to 1 enables the interrupt.

Note: Prior to writing, FLASH_NS/SECCR register must be unlocked as explained in Section 7.6.7.

Table 74 gives a summary of the available flash interface interrupt features. As mentioned in the table below, some flags need to be cleared before a new operation is triggered.

### Table 74. Flash interrupt request

<table>
<thead>
<tr>
<th>Interrupt event event flag</th>
<th>Error flag label in register</th>
<th>Enable control bit</th>
<th>Clear flag to resume operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>End of operation event</td>
<td>EOP</td>
<td>EOPIE</td>
<td>N/A</td>
</tr>
<tr>
<td>Write protection error</td>
<td>WRPERR</td>
<td>WRPERRIE</td>
<td>Yes</td>
</tr>
<tr>
<td>Programming sequence error</td>
<td>PGSERR</td>
<td>PGSERIE</td>
<td>Yes</td>
</tr>
<tr>
<td>Strobe error</td>
<td>STRBERR</td>
<td>STRBERRIE</td>
<td>Yes</td>
</tr>
<tr>
<td>OBK write error</td>
<td>OBKWERR</td>
<td>OBKWERRIE</td>
<td>Yes</td>
</tr>
<tr>
<td>OBK general error</td>
<td>OBKERR</td>
<td>OBKERRIE</td>
<td>Yes</td>
</tr>
<tr>
<td>Inconsistency error</td>
<td>INCERR</td>
<td>INCERRIE</td>
<td>Yes</td>
</tr>
<tr>
<td>Illegal access error</td>
<td>ILAFM</td>
<td>TZEN</td>
<td>No</td>
</tr>
<tr>
<td>Option byte error</td>
<td>OPTCHANGEERR</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>ECC single error correction event</td>
<td>ECCCC</td>
<td>ECCCIE</td>
<td>No</td>
</tr>
<tr>
<td>ECC double error detection event(1)</td>
<td>ECCD</td>
<td>Disabled</td>
<td>No</td>
</tr>
</tbody>
</table>

1. NMI.
The status of the individual maskable interrupt sources described in Table 74 (except for option byte error and ECC) can be read from the FLASH_NS/SECSR register. They can be cleared by setting to 1 the adequate bit in FLASH_NS/SECCCR register.

Note: No unlocking mechanism is required to clear an interrupt.

End of operation event

Setting the end of operation interrupt enable bit (EOPIE) in the FLASH_NS/SECCR register enables the generation of an interrupt at the end of an erase operation, a program operation, or an option byte change.

When managing the OBKey storage area, the EOP is associated also with the OBK swap operation and the alternate area erase operation. EOP bit in the FLASH_NS/SECSR register is also set when one of these events occurs.

Setting CLR_EOP bit to 1 in FLASH_NS/SECCCR register clears EOP flag.
7.11 FLASH registers

Each register is assigned a offset address and a reset value. In case of registers representing option byte value, the reset value is determined by the OBL process. In case of success the reset value is loaded from OB. In case of OBL failure, a highly restrictive default value is set.

7.11.1 FLASH access control register (FLASH_ACR)

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

For more details, refer to Section 7.3.4 and Section 7.3.5.

Address offset: 0x000

Reset value: 0x0000 0013

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PRFTEN**: Prefetch enable. When bit value is modified, user must read back ACR register to be sure PRFTEN has been taken into account.

Bits used to control the prefetch.

0: prefetch disabled.

1: prefetch enabled when latency is at least one wait state.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **WRHIGHFREQ[1:0]**: Flash signal delay

These bits are used to control the delay between nonvolatile memory signals during programming operations. Application software has to program them to the correct value depending on the memory interface frequency. Please refer to Table 37 for details.

*Note: No check is performed to verify that the configuration is correct.*

Two WRHIGHFREQ values can be selected for some frequencies.

Bits 3:0 **LATENCY[3:0]**: Read latency

These bits are used to control the number of wait states used during read operations on both nonvolatile memory banks. The application software has to program them to the correct value depending on the memory interface frequency and voltage conditions.

0000: zero wait states used to read a word from nonvolatile memory

0001: one wait state used to read a word from nonvolatile memory

0010: two wait states used to read a word from nonvolatile memory

... 0111: seven wait states used to read a word from nonvolatile memory

1111: 15 wait states used to read a word from nonvolatile memory

*Note: No check is performed by hardware to verify that the configuration is correct.*
7.11.2  FLASH non-secure key register (FLASH_NSKEYR)

This register is non-secure. It can be written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

FLASH_NSKEYR is a write-only register. The following values must be programmed consecutively to unlock FLASH_NSCR register and allow programming/erasing it. A wrong sequence locks the FLASH_NSCR register until next system reset.

First key = 0x4567 0123
Second key = 0xCDEF 89AB
Address offset: 0x004
Reset value: 0x0000 0000

Bits 31:0  NSKEY[31:0]: Non-volatile memory non-secure configuration access unlock key

7.11.3  FLASH secure key register (FLASH_SECKEYR)

This register is secure. It can be written only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

FLASH_SECKEYR is a write-only register. The following values must be programmed consecutively to unlock FLASH_SECCR register and allow programming/erasing it. A wrong sequence locks the FLASH_SECCR register until next system reset.

First key = 0x4567 0123
Second key = 0xCDEF 89AB
Address offset: 0x008
Reset value: 0x0000 0000

Bits 31:0  SECKEY[31:0]: Non-volatile memory secure configuration access unlock key
7.11.4  **FLASH option key register (FLASH_OPTKEYR)**

This register is non-secure. It can be written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

FLASH_OPTKEYR is a write-only register. The following values must be programmed consecutively to unlock FLASH_OPTCR register. A wrong sequence locks FLASH_OPTCR register until next system reset.

First key = 0x0819 2A3B
Second key = 0x4C5D 6E7F
Address offset: 0x00C
Reset value: 0x0000 0000

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

Bits 31:0  **OPTKEY[31:0]**: FLASH option bytes control access unlock key

7.11.5  **FLASH non-secure OBK key register (FLASH_NSOBKKEYR)**

FLASH_NSOBKKEYR is a write-only register. The following values must be programmed consecutively to unlock FLASH_NSOBKCFGR register. A wrong sequence locks FLASH_NSOBKCFGR register until next system reset.

This register is non-secure. This register can be written only by privileged access.

First key = 0x192A 083B
Second key = 0x6E7F 4C5D
Address offset: 0x0010
Reset value: 0x0000 0000

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

Bits 31:0  **NSOBKKEY[31:0]**: FLASH non-secure option bytes keys control access unlock key
7.11.6 FLASH secure OBK key register (FLASH_SECOBKKEYR)
FLASH_SECOBKKEYR is a write-only register. The following values must be programmed consecutively to unlock FLASH_SECOBKCFGR register. A wrong sequence locks FLASH_SECOBKCFGR register until next system reset.

First key = 0x192A 083B
Second key = 0x6E7F 4C5D

This register is secure. This register can only be written by privileged access.

Address offset: 0x0014
Reset value: 0x0000 0000

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

Bits 31:0 SECOBKKEY[31:0]: FLASH secure option bytes keys control access unlock key

7.11.7 FLASH operation status register (FLASH_OPSR)
This register is non-secure. This register can be read by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x0018
Reset value: 0xXXXX XXXX

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

ADDR_OP[15:0]
7.11.8 FLASH option control register (FLASH_OPTCR)

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Access: No wait state when no FLASH memory operation is ongoing. The FLASH_OPTCR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it while the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x01C

Reset value: 0x00000001

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>SWAP_BANK</th>
<th>Bank swapping option configuration bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPT</td>
<td>STRT</td>
<td>LOCK</td>
</tr>
</tbody>
</table>

Bits 31:29 CODE_OP[2:0]: Flash memory operation code
- 000: No flash operation on going during previous reset
- 001: Single write operation interrupted
- 010: OBK alternate sector erase
- 011: Sector erase operation interrupted
- 100: Bank erase operation interrupted
- 101: Mass erase operation interrupted
- 110: Option change operation interrupted
- 111: OBK swap sector request

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 OTP_OP: OTP operation interrupted
Indicates that reset interrupted an ongoing operation in OTP area (or OBKeys area).

Bit 23 SYSF_OP: Operation in system flash memory interrupted
Indicates that reset interrupted an ongoing operation in system flash.

Bit 22 BK_OP: Interrupted operation bank
It indicates which bank was concerned by operation.

Bit 21 DATA_OP: Flash high-cycle data area operation interrupted
It indicates if flash high-cycle data area is concerned by operation.

Bit 20 Reserved, must be kept at reset value.

Bits 19:0 ADDR_OP[19:0]: Interrupted operation address
7.11.9 FLASH non-secure status register (FLASH_NSSR)

This register is non-secure. It can be read by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x20

Reset value: 0x0000 000X

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

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 OPTCHANGEERR: Option byte change error flag

OPTCHANGEERR flag indicates that an error occurred during an option byte change operation. When OPTCHANGEERR is set to 1, the option byte change operation did not successfully complete. An interrupt is generated when this flag is raised if the OPTCHANGEERRIE bit of FLASH_NSSCR register is set to 1.

Writing 1 to CLR_OPTCHANGEERR of register FLASH_NSSCR clears OPTCHANGEERR.

0: no option byte change errors occurred
1: one or more errors occurred during an option byte change operation.

Note: The OPTSTRT bit in FLASH_OPTCR cannot be set while OPTCHANGEERR is set.
Bit 22 **OBKWERR**: OBK write error flag

OBKWERR flag is raised when the address is not virgin on a write access to the OBK storage. Alternatively also when the OBK selector in the alternate sector is not virgin during a swap operation.

0: no OBK write error occurred
1: an OBK write error occurred

Bit 21 **OBKERR**: OBK general error flag

OBKERR flag is raised when the OBK-HDPL signal from the SBS does not match the HDPL value associated with the key slot during access to the key location. Alternatively also when the ALT_SECT is unexpectedly changed while the write buffer is being filled.

0: no OBK general error occurred
1: an OBK general error occurred

Bit 20 **INCERR**: inconsistency error flag

INCERR flag is raised when a inconsistency error occurs. An interrupt is generated if INCERRIE is set to 1. Writing 1 to CLR_INCERR bit in the FLASH_NSCCR register clears INCERR.

0: no inconsistency error occurs
1: a inconsistency error occurs

Bit 19 **STRBERR**: strobe error flag

STRBERR flag is raised when a strobe error occurs (when the master attempts to write several times the same byte in the write buffer). An interrupt is generated if the STRBERR bit is set to 1. Writing 1 to CLR_STRBERR bit in FLASH_NSCCR register clears STRBERR.

0: no strobe error occurred
1: a strobe error occurred

Bit 18 **PGSERR**: programming sequence error flag

PGSERR flag is raised when a sequence error occurs. An interrupt is generated if the PGSERRIE bit is set to 1. Writing 1 to CLR_PGSERR bit in FLASH_NSCCR register clears PGSERR.

0: no sequence error occurred
1: a sequence error occurred

Bit 17 **WRPERR**: write protection error flag

WRPERR flag is raised when a protection error occurs during a program operation. An interrupt is also generated if the WRPERRIE is set to 1. Writing 1 to CLR_WRPERR bit in FLASH_NSCCR register clears WRPERR.

0: no write protection error occurred
1: a write protection error occurred

Bit 16 **EOP**: end of operation flag

EOP flag is set when a operation (program/erase) completes. An interrupt is also generated if the EOPIE is set to 1. It is not necessary to reset EOP before starting a new operation. EOP bit is cleared by writing 1 to CLR_EOP bit in FLASH_NSCCR register.

0: no operation completed
1: a operation completed

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **DBNE**: data buffer not empty flag

DBNE flag is set when the flash interface is processing 6-bits ECC data in dedicated buffer. This bit cannot be set to 0 by software. The hardware resets it once the buffer is free.

0: data buffer not used
1: data buffer used, wait
Bit 2  Reserved, must be kept at reset value.

Bit 1  **WBNE**: write buffer not empty flag

WBNE flag is set when the flash interface is waiting for new data to complete the write buffer. In this state, the write buffer is not empty. WBNE is reset by hardware each time the write buffer is complete or the write buffer is emptied following one of the event below:
- the application software forces the write operation using FW bit in FLASH_NS
- the memory detects an error that involves data loss

This bit cannot be reset by software writing 0 directly. To reset it, clear the write buffer by performing any of the above listed actions, or send the missing data.
0: write buffer empty or full
1: write buffer waiting data to complete

Bit 0  **BSY**: busy flag

BSY flag indicates that a flash memory is busy by an operation (write, erase, option byte change, OBK operation). It is set at the beginning of a flash memory operation and cleared when the operation finishes, or an error occurs.
0: no programming, erase or option byte change operation being executed
1: programming, erase or option byte change operation being executed

### 7.11.10 FLASH secure status register (FLASH_SECSR)

This register is secure. It can be read only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x024

Reset value: 0x0000 000X

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

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

Bit 22  **OBKWERR**: OBK write error flag

OBKWERR flag is raised when the address is not virgin on a write access to the OBK storage. Alternatively also when the OBK selector in the alternate sector is not virgin during a swap operation.
0: no OBK write error occurred
1: an OBK write error occurred

Bit 21  **OBKERR**: OBK general error flag

OBKERR flag is raised when the OBK-HDPL signal from the SBS does not match the HDPL value associated with the key slot during access to the key location. Alternatively also when the ALT_SECT is unexpectedly changed while the write buffer is being filled.
0: no OBK general error occurred
1: an OBK general error occurred
Bit 20 **INCERR**: inconsistency error flag

INCERR flag is raised when an inconsistency error occurs. An interrupt is generated if INCERRIE is set to 1. Writing 1 to CLR_INCERR bit in the FLASH_SECCCR register clears INCERR.

0: no inconsistency error occurred
1: an inconsistency error occurred

Bit 19 **STRBERR**: strobe error flag

STRBERR flag is raised when a strobe error occurs (when the master attempts to write several times the same byte in the write buffer). An interrupt is generated if the STRBERRIE bit is set to 1. Writing 1 to CLR_STRBERR bit in FLASH_SECCCR register clears STRBERR.

0: no strobe error occurred
1: a strobe error occurred

Bit 18 **PGSERR**: programming sequence error flag

PGSERR flag is raised when a sequence error occurs. An interrupt is generated if the PGSERRIE bit is set to 1. Writing 1 to CLR_PGSERR bit in FLASH_SECCCR register clears PGSERR.

0: no sequence error occurred
1: a sequence error occurred

Bit 17 **WRPERR**: write protection error flag

WRPERR flag is raised when a protection error occurs during a program operation. An interrupt is also generated if the WRPERRIE is set to 1. Writing 1 to CLR_WRPERR bit in FLASH_SECCCR register clears WRPERR.

0: no write protection error occurred
1: a write protection error occurred

Bit 16 **EOP**: end of operation flag

EOP flag is set when an operation (program/erase) completes. An interrupt is generated if the EOPIE is set to 0. It is not necessary to reset EOP before starting a new operation. EOP bit is cleared by writing 1 to CLR_EOP bit in FLASH_SECCCR register.

0: no operation completed
1: an operation completed

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **DBNE**: data buffer not empty flag

DBNE flag is set when the memory interface is processing 6-bits ECC data in dedicated buffer. This bit cannot be set to 0 by software. The hardware resets it once the buffer is free.

0: data buffer not used
1: data buffer used, wait

Bit 2 Reserved, must be kept at reset value.

Bit 1 **WBNE**: write buffer not empty flag

WBNE flag is set when the flash interface is waiting for new data to complete the write buffer. In this state, the write buffer is not empty. WBNE is reset by hardware each time the write buffer is complete or the write buffer is emptied following one of the event below:
- the application software forces the write operation using FW bit in FLASH_SECCR
- the flash interface detects an error that involves data loss

This bit cannot be reset by writing 0 directly by software. To reset it, clear the write buffer by performing any of the above listed actions, or send the missing data.

0: write buffer empty or full
1: write buffer waiting data to complete
### Bit 0 BSY: busy flag
BSY flag indicates that a FLASH memory is busy (write, erase, option byte change, OBK operations). It is set at the beginning of a flash memory operation and cleared when the operation finishes or an error occurs.
- 0: no programming, erase or option byte change operation being executed
- 1: programming, erase or option byte change operation being executed

### 7.11.11 FLASH non-secure control register (FLASH_NSCR)

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Access: No wait state when no FLASH memory operation is ongoing. The FLASH_NSCR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x028
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rs</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MER</td>
<td>Res.</td>
<td>Res.</td>
<td>SNB[6:0]</td>
<td>STRT</td>
<td>FW</td>
<td>BER</td>
<td>SER</td>
<td>PG</td>
<td>LOCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rs</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Bit 31 BKSEL: Bank selector bit
- BKSEL can only be programmed when LOCK is cleared to 0. The bit selects physical bank, SWAP_BANK setting is ignored.
- 0: Bank1 is selected for Bank erase / sector erase / interrupt enable
- 1: Bank2 is selected for BER / SER

#### Bits 30:24 Reserved, must be kept at reset value.

#### Bit 23 OPTCHANGEERRIE: Option byte change error interrupt enable bit
- This bit controls if an interrupt must be generated when an error occurs during an option byte change. It can be programmed only when LOCK bit is cleared to 0.
- 0: no interrupt is generated when an error occurs during an option byte change
- 1: an interrupt is generated when and error occurs during an option byte change.

#### Bit 22 OBKWERRIE: OBK write error interrupt enable bit
- OBKWERRIE enables generation of interrupt in case of OBK specific write error. This bit can be programmed only when LOCK bit is cleared to 0.
- 0: no interrupt is generated on OBK write error
- 1: an interrupt is generated on OBK write error

#### Bit 21 OBKERRIE: OBK general error interrupt enable bit
- OBKERRIE enables generating an interrupt in case of OBK specific access error. This bit can be programmed only when LOCK bit is cleared to 0.
- 0: no interrupt is generated on OBK general access error
- 1: an interrupt is generated on OBK general access error
Bit 20  **INCERRIE**: inconsistency error interrupt enable bit
When INCERRIE bit is set to 1, an interrupt is generated when an inconsistency error occurs during a write operation. INCERRIE can be programmed only when LOCK is cleared to 0.
0: no interrupt generated when a inconsistency error occurs
1: interrupt generated when a inconsistency error occurs.

Bit 19  **STRBERRIE**: strobe error interrupt enable bit
When STRBERRIE bit is set to 1, an interrupt is generated when a strobe error occurs (the master programs several times the same byte in the write buffer) during a write operation. STRBERRIE can be programmed only when LOCK is cleared to 0.
0: no interrupt generated when a strobe error occurs
1: interrupt generated when strobe error occurs.

Bit 18  **PGSERRIE**: programming sequence error interrupt enable bit
When this bit is set to 1, an interrupt is generated when a sequence error occurs during a program operation. PGSERRIE can be programmed only when LOCK is cleared to 0.
0: no interrupt generated when a sequence error occurs
1: interrupt generated when sequence error occurs.

Bit 17  **WRPERRIE**: write protection error interrupt enable bit
When this bit is set to 1, an interrupt is generated when a protection error occurs during a program operation. WRPERRIE can be programmed only when LOCK is cleared to 0.
0: no interrupt generated when a protection error occurs
1: interrupt generated when a protection error occurs.

Bit 16  **EOPIE**: end of operation interrupt control bit
Setting EOPIE bit to 1 enables the generation of an interrupt at the end of a program or erase operation. EOPIE can be programmed only when LOCK is cleared to 0.
0: no interrupt generated at the end of operation.
1: interrupt enabled when at the end of operation.

Bit 15  **MER**: Mass erase request
Setting MER bit to 1 requests a mass erase operation (user flash memory only). MER can be programmed only when LOCK is cleared to 0.
If BER or SER are both set, a PGSERR is raised.
0: mass erase not requested
1: mass erase requested

*Error is triggered when a mass erase is required and some sectors are protected.*

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:6  **SNB[6:0]**: sector erase selection number
These bits are used to select the target sector for an erase operation (they are unused otherwise). SNB can be programmed only when LOCK is cleared to 0.
0x00: Sector 0 selected
0x01: Sector 1 selected
...
0x7F: Sector 127 selected

Bit 5  **STRT**: erase start control bit
STRT bit is used to start a sector erase or a bank erase operation. STRT can be programmed only when LOCK is cleared to 0.
STRT is reset at the end of the operation or when an error occurs. It cannot be reseted by software.
Bit 4  **FW**: write forcing control bit
FW forces a write operation even if the write buffer is not full. In this case all bits not written are set to 1 by hardware. FW can be programmed only when LOCK is cleared to 0.
The memory resets FW when the corresponding operation has been acknowledged.
**Note**: Using a force-write operation prevents the application from updating later the missing bits with something else than 1, because it is likely that it leads to permanent ECC error.
Write forcing is effective only if the write buffer is not empty and was filled by non-secure access (in particular, FW does not start several write operations when the force-write operations are performed consecutively).
Since there is just one write buffer, FW can force a write in bank1 or bank2.

Bit 3  **BER**: erase request
Setting BER bit to 1 requests a bank erase operation (user flash memory only). BER can be programmed only when LOCK is cleared to 0.
If MER and SER are also set, a PGSERR is raised.
0: bank erase not requested
1: bank erase requested

**Note**: Write protection error is triggered when a bank erase is required and some sectors are protected.

Bit 2  **SER**: sector erase request
Setting SER bit to 1 requests a sector erase. SER can be programmed only when LOCK is cleared to 0.
If MER and SER are also set, a PGSERR is raised.
0: sector erase not requested
1: sector erase requested

Bit 1  **PG**: programming control bit
PG can be programmed only when LOCK is cleared to 0.
PG allows programming in Bank1 and Bank2.
0: programming disabled
1: programming enabled

Bit 0  **LOCK**: configuration lock bit
This bit locks the FLASH_NSCR register. The correct write sequence to FLASH_NSKEYR register unlocks this bit. If a wrong sequence is executed, or if the unlock sequence to FLASH_NSKEYR is performed twice, this bit remains locked until the next system reset.
LOCK can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK changes from 0 to 1, the other bits of FLASH_NSCR register do not change.
0: FLASH_NSCR register unlocked
1: FLASH_NSCR register locked
7.11.12  **FLASH secure control register (FLASH_SECCCR)**

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Access: No wait state when no FLASH memory operation is ongoing. The FLASH_SECCCR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x02C

Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>BKSEL: bank selector bit</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>INV: Flash memory security state invert.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>OBKWERRIE: OBK write error interrupt enable bit</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>OBKERRIE: OBK general error interrupt enable bit</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>INCERRIE: inconsistency error interrupt enable bit</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>STRT</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>SNB[6:0]</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>FW</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>BER</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>SER</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>PG</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>LOCK</td>
<td>rs</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td>Res.</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31  **BKSEL**: bank selector bit

- BKSEL can only be programmed when LOCK is cleared to 0. The bit selects physical bank, SWAP_BANK setting is ignored.
- 0: Bank1 is selected for Bank erase / sector erase / interrupt enable
- 1: Bank2 is selected for BER / SER

Bit 30  Reserved, must be kept at reset value.

Bit 29  **INV**: Flash memory security state invert.

- This bit inverts the flash memory security state.

Bits 28:23  Reserved, must be kept at reset value.

Bit 22  **OBKWERRIE**: OBK write error interrupt enable bit

- OBKWERRIE enables generation of interrupt in case of OBK specific write error.
- OBKWERRIE can be programmed only when LOCK is cleared to 0.
- 0: no interrupt is generated on OBK write error
- 1: an interrupt is generated on OBK write error

Bit 21  **OBKERRIE**: OBK general error interrupt enable bit

- OBKERRIE enables generating an interrupt in case of OBK specific access error.
- OBKERRIE can be programmed only when LOCK is cleared to 0.
- 0: no interrupt is generated on OBK general access error
- 1: an interrupt is generated on OBK general access error

Bit 20  **INCERRIE**: inconsistency error interrupt enable bit

- When INCERRIE bit is set to 1, an interrupt is generated when an inconsistency error occurs during a write operation. INCERRIE can be programmed only when LOCK is cleared to 0.
- 0: no interrupt generated when a inconsistency error occurs
- 1: interrupt generated when a inconsistency error occurs.
Bit 19 **STRBERRIE**: strobe error interrupt enable bit
When STRBERRIE bit is set to 1, an interrupt is generated when a strobe error occurs (the master programs several times the same byte in the write buffer) during a write operation. STRBERRIE can be programmed only when LOCK is cleared to 0.
- 0: no interrupt generated when a strobe error occurs
- 1: interrupt generated when strobe error occurs

Bit 18 **PGSERRIE**: programming sequence error interrupt enable bit
When PGSERRIE bit is set to 1, an interrupt is generated when a sequence error occurs during a program operation. PGSERRIE can be programmed only when LOCK is cleared to 0.
- 0: no interrupt generated when a sequence error occurs
- 1: interrupt generated when sequence error occurs

Bit 17 **WRPERRIE**: write protection error interrupt enable bit
When WRPERRIE bit is set to 1, an interrupt is generated when a protection error occurs during a program operation. WRPERRIE can be programmed only when LOCK is cleared to 0.
- 0: no interrupt generated when a protection error occurs
- 1: interrupt generated when a protection error occurs

Bit 16 **EOPIE**: end of operation interrupt control bit
Setting EOPIE bit to 1 enables the generation of an interrupt at the end of a program/erase operation. EOPIE can be programmed only when LOCK is cleared to 0.
- 0: no interrupt generated at the end of operation.
- 1: interrupt enabled when at the end of operation

Bit 15 **MER**: mass erase request
Setting MER bit to 1 requests a mass erase operation (user flash memory only). MER can be programmed only when LOCK is cleared to 0.
- 0: mass erase not requested
- 1: mass erase requested

*Error is triggered when a mass erase is required and some sectors are protected.*

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:6 **SNB[6:0]**: sector erase selection number
These bits are used to select the target sector for an erase operation (they are unused otherwise). SNB can be programmed only when LOCK is cleared to 0.
- 0x00: Sector 0 selected
- 0x01: Sector 1 selected
- ..
- 0x7F: Sector 127 selected

Bit 5 **STRT**: erase start control bit
STRT bit is used to start a sector erase or a bank erase operation. STRT can be programmed only when LOCK is cleared to 0.
STRT is reseted at the end of the operation or when an error occurs. It cannot be reset by software.
Bit 4 **FW**: write forcing control bit

FW forces a write operation even if the write buffer is not full. In this case all bits not written are set to 1 by hardware. FW can be programmed only when LOCK is cleared to 0.
The memory resets FW when the corresponding operation has been acknowledged.

*Note:* Using a force-write operation prevents the application from updating later the missing bits with something else than 1, because it is likely that it leads to permanent ECC error.

Write forcing is effective only if the write buffer is not empty and was filled by secure access (in particular, FW does not start several write operations when the force-write operations are performed consecutively).

Since there is just one write buffer, FW can force a write in bank1 or bank2.

Bit 3 **BER**: erase request

Setting BER bit to 1 requests a bank erase operation (user flash memory only). BER can be programmed only when LOCK is cleared to 0.

If MER and SER are also set, a PGSERR is raised.

0: bank erase not requested
1: bank erase requested

*Note:* Write protection error is triggered when a bank erase is required and some sectors are protected.

Bit 2 **SER**: sector erase request

Setting SER bit to 1 requests a sector erase. SER can be programmed only when LOCK is cleared to 0.

If BER and MER are also set, a PGSERR is raised.

0: sector erase not requested
1: sector erase requested

Bit 1 **PG**: programming control bit

PG can be programmed only when LOCK is cleared to 0.

PG allows programming in Bank1 and Bank2.

0: programming disabled
1: programming enabled

Bit 0 **LOCK**: configuration lock bit

This bit locks the FLASH_SECCR register. The correct write sequence to FLASH_SECKEYR register unlocks this bit. If a wrong sequence is executed, or if the unlock sequence to FLASH_NSKEYR is performed twice, this bit remains locked until the next system reset.

LOCK can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK changes from 0 to 1, the other bits of FLASH_SECCR register do not change.

0: FLASH_SECCR register unlocked
1: FLASH_SECCR register locked
7.11.13  FLASH non-secure clear control register (FLASH_NSCCR)

This register is non-secure. It can be written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x030

Reset value: 0x0000 0000

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

Bits 31:24  Reserved, must be kept at reset value.

- **Bit 23  CLR_OPTCHANGEERR**: Clear the flag corresponding flag in FLASH_NSSR by writing this bit.
- **Bit 22  CLR_OBKWERR**: OBKWERR flag clear bit.
  Setting this bit to 1 resets to 0 OBKWERR flag in FLASH_NSSR register.
- **Bit 21  CLR_OBKERR**: OBKERR flag clear bit.
  Setting this bit to 1 resets to 0 OBKERR flag in FLASH_NSSR register.
- **Bit 20  CLR_INCERR**: INCERR flag clear bit
  Setting this bit to 1 resets to 0 INCERR flag in FLASH_NSSR register.
- **Bit 19  CLR_STRBERR**: STRBERR flag clear bit
  Setting this bit to 1 resets to 0 STRBERR flag in FLASH_NSSR register.
- **Bit 18  CLR_PGSERR**: PGSEERR flag clear bit
  Setting this bit to 1 resets to 0 PGSEERR flag in FLASH_NSSR register.
- **Bit 17  CLR_WRPERR**: WRPERR flag clear bit
  Setting this bit to 1 resets to 0 WRPERR flag in FLASH_NSSR register.
- **Bit 16  CLR_EOP**: EOP flag clear bit
  Setting this bit to 1 resets to 0 EOP flag in FLASH_NSSR register.

Bits 15:0  Reserved, must be kept at reset value.
7.11.14  **FLASH secure clear control register (FLASH_SECCCR)**

This register is secure. It can be written only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x034

Reset value: 0x0000 0000

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

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

Bit 22  **CLR_OBKWERR**: OBKWERR flag clear bit

Setting this bit to 1 resets to 0 OBKWERR flag in FLASH_SECSR register.

Bit 21  **CLR_OBKERR**: OBKWERR flag clear bit

Setting this bit to 1 resets to 0 OBKWERR flag in FLASH_SECSR register.

Bit 20  **CLR_INCERR**: INCERR flag clear bit

Setting this bit to 1 resets to 0 INCERR flag in FLASH_SECSR register.

Bit 19  **CLR_STRBERR**: STRBERR flag clear bit

Setting this bit to 1 resets to 0 STRBERR flag in FLASH_SECSR register.

Bit 18  **CLR_PGSERR**: PGSERR flag clear bit

Setting this bit to 1 resets to 0 PGSERR flag in FLASH_SECSR register.

Bit 17  **CLR_WRPERR**: WRPERR flag clear bit

Setting this bit to 1 resets to 0 WRPERR flag in FLASH_SECSR register.

Bit 16  **CLR_EOP**: EOP flag clear bit

Setting this bit to 1 resets to 0 EOP flag in FLASH_SECSR register.

Bits 15:0  Reserved, must be kept at reset value.
### 7.11.15 FLASH privilege configuration register (FLASH_PRIVCFGR)

Address offset: 0x03C

Reset value: 0x0000 0000

This register can be read by both privileged and unprivileged access. NSPRIV is a non-secure bit. SPRIV is a secure bit. It can only be written by privilege mode.

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

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

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

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

Bit 1 **NSPRIV**: privilege attribute for non-secure registers
- 0: access to non-secure registers is always granted
- 1: access to non-secure registers is denied in case of unprivileged access.

Bit 0 **SPRIV**: privilege attribute for secure registers
- 0: access to secure registers is always granted
- 1: access to secure registers is denied in case of unprivileged access.

### 7.11.16 FLASH non-secure OBK configuration register (FLASH_NSOBKCFGR)

Register is only available when TZ_STATE = 0xC3. This register is non-secure. It can be read and written by both secure and non-secure access. This register can only be accessed by privilege code.

The FLASH_NSOBKCFGR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x040

Reset value: 0x01FF 0000

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

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>ALT_SECT</th>
<th>ALT_SECT</th>
<th>SWAP_SECT_REQ</th>
<th>LOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.
Bits 24:16 **SWAP_OFFSET[8:0]**: Key index (offset /16 bits) pointing for next swap.
- 0x00: means that no OBK is copied from current to alternate OBK sector during SWAP operation.
- 0x01 means that only the first OBK data (128 bits) is copied from current to alternate OBK sector
- 0x02 means that the two first OBK data is copied ...
  ...
- 0x1FF: means that all OBK data (511) is copied

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **ALT_SECT_ERASE**: alternate sector erase bit
- 0: do not touch OBK sector
- 1: erase the alternate OBK sector
  When ALT_SECT bit is set, use this bit to generate an erase command for the OBK alternate sector. It is set only by Software and cleared when the OBK swap operation is completed or an error occurs (PGSER). It is reseted at the same time as BUSY bit.

Bit 2 **ALT_SECT**: alternate sector bit
- 0: current OBK sector is mapped to OBK address range for access
- 1: alternate OBK sector is mapped to OBK address range for access
  This bit must not change while filling the write buffer, otherwise an error (OBKERR) is generated

Bit 1 **SWAP_SECT_REQ**: OBK swap sector request bit
- 0: no swap requested
- 1: launch the sector swap
  When set, all the OBKs which have not been updated in the alternate sector is copied from current sector to alternate one.
  The SWAP_OFFSET value must be a certain minimum value in order for the swap to be launched in OBK-HDPL ≠ 0. Minimum value is 16 for OBK-HDPL = 1, 144 for OBK-HDPL = 2 and 192 for OBK-HDPL = 3.

Bit 0 **LOCK**: OBKCFGR lock option configuration bit
- This bit locks the FLASH_NSOBKCFGR register. The correct write sequence to FLASH_NSOBKKEYR register unlocks this bit. If a wrong sequence is executed, or if the unlock sequence to FLASH_NSOBKKEYR is performed twice, this bit remains locked until the next system reset. LOCK can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK changes from 0 to 1, the other bits of FLASH_NSCR register do not change.
- 0: FLASH_NSOBKCFGR register unlocked
- 1: FLASH_NSOBKCFGR register locked
### 7.11.17 FLASH secure OBK configuration register (FLASH_SECObKCFGR)

This register can be accessed only in secure privilege mode. The FLASH_SECObKCFGR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

**Address offset:** 0x044  
**Reset value:** 0x01FF 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **SWAP_OFFSET[8:0]:** key index (offset /16 bits) pointing for next swap.  
- 0x0: no OBK is copied from current to alternate OBK sector during SWAP operation.  
- 0x1: only the first OBK data (128 bits) is copied from current to alternate OBK sector  
- 0x2: the two first OBK data are copied …  
  
…  
- 0x1FF: that the 511 first OBK data are copied

Bits 15:4 Reserved, must be kept at reset value.

**Bit 3** **AL_T_SECT_ERASE:** alternate sector erase bit  
0: do not touch OBK sector  
1: erase the alternate OBK sector  
When AL_T_SECT bit is set, use this bit to generate an erase command for the OBK alternate sector. It is set only by Software and cleared when the OBK swap operation is completed or an error occurs (PGSERR). It is reseted at the same time as the BUSY bit.

**Bit 2** **AL_T_SECT:** alternate sector bit  
0: current OBK sector is mapped to OBK address range for access  
1: alternate OBK sector is mapped to OBK address range for access  
This bit must not change while filling the write buffer, otherwise an error is generated

**Bit 1** **SWAP_SECT_REQ:** OBK swap sector request bit  
0: no swap requested  
1: launch the sector swap  
When set, all the OBKs which have not been updated in the alternate sector is copied from current sector to alternate one.

The **SWAP_OFFSET** value must be a certain minimum value in order for the swap to be launched in OBK-HDPL ≠ 0. Minimum value is 16 for OBK-HDPL = 1, 144 for OBK-HDPL = 2 and 192 for OBK-HDPL = 3.
Bit 0 **LOCK**: OBKCFGR lock option configuration bit

This bit locks the FLASH_OBKCFGR register. The correct write sequence to FLASH_SECOBKKEYR register unlocks this bit. If a wrong sequence is executed, or if the unlock sequence to FLASH_SECOBKKEYR is performed twice, this bit remains locked until the next system reset. LOCK can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK changes from 0 to 1, the other bits of FLASH_NSCR register do not change.

0: FLASH_OBKCFGR register unlocked
1: FLASH_OBKCFGR register locked

### 7.11.18 FLASH HDP extension register (FLASH_HDPEXTR)

All bits in this register are non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

The register can only be modified in HDPL<=2.

The values in this register cannot be decremented. Attempt to write a lower value that current is ignored.

Address offset: 0x048
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-23</th>
<th>Bit 22-16</th>
<th>Bit 15-7</th>
<th>Bit 6-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>HDP2_EXT[6:0]</td>
<td>rw rw rw rw rw rw rw rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>HDP1_EXT[6:0]</td>
<td>rw rw rw rw rw rw rw rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 22:16 **HDP2_EXT[6:0]**: HDP area extension in 8 Kbytes sectors in bank 2. Extension is added after the HDP2_END sector (included).

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

Bits 6:0 **HDP1_EXT[6:0]**: HDP area extension in 8 Kbytes sectors in Bank1. Extension is added after the HDP1_END sector (included).

### 7.11.19 FLASH option status register (FLASH_OPTSR_CUR)

This register is non-secure. It can be read by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Default value: 0x0030 5CD8 (value in case of double ECC issue during OBL).

This read-only register reflects the current values of corresponding option bits.

Address offset: 0x050
Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)
Bit 31  **SWAP_BANK**: Bank swapping option status bit
SWAP_BANK reflects whether Bank1 and Bank2 are swapped or not.
SWAP_BANK is loaded to SWAP_BANK of FLASH_OPTCR after a reset.
0: Bank1 and Bank2 not swapped
1: Bank1 and Bank2 swapped

Bit 30 Reserved, must be kept at reset value.

Bits 29:22  **BOOT_UBE[7:0]**: Available only on cryptography enabled devices.
Unique boot entry control, selects either ST or OEM iRoT for secure boot.
- 0xC3: ST-iRoT (system flash) selected
- 0xB4: OEM-iRoT (user flash) selected. In Open PRODUCT_STATE this value selects bootloader. Default value.

Bit 21  **IWDG_STDBY**: IWDG Standby mode freeze option status bit
When set the independent watchdog IWDG is frozen in system Standby mode.
0: Independent watchdog frozen in Standby mode
1: Independent watchdog keep running in Standby mode.

Bit 20  **IWDG_STOP**: IWDG Stop mode freeze option status bit
When set the independent watchdog IWDG is in system Stop mode.
0: Independent watchdog frozen in system Stop mode
1: Independent watchdog keep running in system Stop mode.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17  **IO_VDDIO2_HSLV**: High-speed IO at low $V_{DDIO2}$ voltage configuration bit.
This bit can be set only with $V_{DDIO2}$ below 2.7 V.
- 0: High-speed IO at low $V_{DDIO2}$ voltage feature disabled ($V_{DDIO2}$ can exceed 2.7 V)
- 1: High-speed IO at low $V_{DDIO2}$ voltage feature enabled ($V_{DDIO2}$ remains below 2.7 V)

Bit 16  **IO_VDD_HSLV**: High-speed IO at low $V_{DD}$ voltage configuration bit.
This bit can be set only with $V_{DD}$ below 2.7 V.
- 0: High-speed IO at low $V_{DD}$ voltage feature disabled ($V_{DD}$ can exceed 2.7 V)
- 1: High-speed IO at low $V_{DD}$ voltage feature enabled ($V_{DD}$ remains below 2.7 V)

Bits 15:8  **PRODUCT_STATE[7:0]**: Life state code (based on Hamming 8,4). More information in
Section 7.6.11: Product state transitions.

Bit 7  **NRST_STDBY**: Core domain Standby entry reset option status bit
0: a reset is generated when entering Standby mode on core domain
1: no reset generated when entering Standby mode on core domain.

Bit 6  **NRST_STOP**: Core domain Stop entry reset option status bit
0: a reset is generated when entering Stop mode on core domain
1: no reset generated when entering Stop mode on core domain.

Bit 5 Reserved, must be kept at reset value.
Bit 4  **WWDG_SW**: WWDG control mode option status bit  
   0: WWDG watchdog is controlled by hardware  
   1: WWDG watchdog is controlled by software  

Bit 3  **IWDG_SW**: IWDG control mode option status bit  
   0: IWDG watchdog is controlled by hardware  
   1: IWDG watchdog is controlled by software  

Bit 2  **BORH_EN**: Brownout high enable  
   0: disable  
   1: enable  

Bits 1:0  **BOR_LEV[1:0]**: Brownout level option status bit  
   These bits reflects the power level that generates a system reset.  
   00 or 11: BOR Level 1, the threshold level is low (around 2.1 V)  
   01: BOR Level 2, the threshold level is medium (around 2.4 V)  
   10: BOR Level 3, the threshold level is high (around 2.7 V)  

### 7.11.20  FLASH option status register (FLASH_OPTSR_PRG)  

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.  

This register is used to program values in corresponding option bits. Values after reset reflects the current values of the corresponding option bits.  

Address offset: 0x054  
Reset value: 0xXXXX XXXX  
(Register bits 0 to 31 are loaded with values from the flash memory at OBL)  

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

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31  **SWAP_BANK**: Bank swapping option configuration bit  
   SWAP_BANK option bit is used to configure whether the Bank1 and Bank2 are swapped or not. This bit is loaded with the SWAP_BANK bit of FLASH_OPTSR_CUR register after a reset.  
   0: Bank1 and Bank2 not swapped  
   1: Bank1 and Bank2 swapped  

Bit 30  Reserved, must be kept at reset value.
Embedded flash memory (FLASH)  RM0481

Bits 29:22  **BOOT_UBE[7:0]**: Available only on cryptography enabled devices. Unique boot entry control, selects either ST or OEM iRoT for secure boot.
0xC3: ST-iRoT (system flash) selected
0xB4: OEM-iRoT (user flash) selected.
In Open PRODUCT_STATE this value selects bootloader. Default value.

**Bit 21  IWDG_STDBY**: IWDG Standby mode freeze option status bit
When set the independent watchdog IWDG is frozen in system Standby mode.
0: Independent watchdog frozen in Standby mode
1: Independent watchdog keep running in Standby mode.

**Bit 20  IWDG_STOP**: IWDG Stop mode freeze option status bit
When set the independent watchdog IWDG is in system Stop mode.
0: Independent watchdog frozen in system Stop mode
1: Independent watchdog keep running in system Stop mode.

Bits 19:18  Reserved, must be kept at reset value.

**Bit 17  IO_VDDIO2_HSLV**: High-speed IO at low VDDIO2 voltage configuration bit.
This bit can be set only with VDDIO2 below 2.7 V.
0: High-speed IO at low VDDIO2 voltage feature disabled (VDDIO2 can exceed 2.7 V)
1: High-speed IO at low VDDIO2 voltage feature enabled (VDDIO2 remains below 2.7 V)

**Bit 16  IO_VDD_HSLV**: High-speed IO at low VDD voltage configuration bit.
This bit can be set only with VDD below 2.7 V.
0: High-speed IO at low VDD voltage feature disabled (VDD can exceed 2.7 V)
1: High-speed IO at low VDD voltage feature enabled (VDD remains below 2.7 V)

**Bits 15:8  PRODUCT_STATE[7:0]**: Life state code (based on Hamming 8,4). More information in Section 7.6.11: Product state transitions.

**Bit 7  NRST_STDBY**: Core domain Standby entry reset option configuration bit
0: a reset is generated when entering Standby mode on core domain
1: no reset generated when entering Standby mode on core domain.

**Bit 6  NRST_STOP**: Core domain Stop entry reset option configuration bit
0: a reset is generated when entering Stop mode on core domain
1: no reset generated when entering Stop mode on core domain.

**Bit 5  Reserved, must be kept at reset value.**

**Bit 4  WWDG_SW**: WWGD control mode option configuration bit
0: WWGD watchdog is controlled by hardware
1: WWGD watchdog is controlled by software

**Bit 3  IWDG_SW**: IWDG control mode option configuration bit
0: IWDG watchdog is controlled by hardware
1: IWDG watchdog is controlled by software

**Bit 2  BORH_EN**: Brownout high enable configuration bit
0: disable
1: enable

**Bits 1:0  BOR_LEV[1:0]**: Brownout level option configuration bit
These bits reflects the power level that generates a system reset.
00 or 11: BOR Level 1, the threshold level is low (around 2.1 V)
01: BOR Level 2, the threshold level is medium (around 2.4 V)
10: BOR Level 3, the threshold level is high (around 2.7 V)
7.11.21  **FLASH non-secure EPOCH register (FLASH_NSEPOCHR_CUR)**

This register is non-secure. It can be read by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of corresponding option bits.

Address offset: 0x060

Reset value: 0xXXXX XXXX

:Register bits 0 to 31 are loaded with values from the flash memory at OBL

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

Bits 31:24  Reserved, must be kept at reset value.

Bits 23:0  **NS_EPOCH[23:0]**: Non-volatile non-secure EPOCH counter

7.11.22  **FLASH secure EPOCH register (FLASH_SECEPOCHR_CUR)**

This register is non-secure. It can be read by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of corresponding option bits.

Address offset: 0x068

Reset value: 0xXXXX XXXX

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

Bits 31:24  Reserved, must be kept at reset value.

Bits 23:0  **SEC_EPOCH[23:0]**: Non-volatile secure EPOCH counter
7.11.23  **FLASH option status register 2 (FLASH_OPTSR2_CUR)**

This register is non-secure. It can be read by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Default value: 0xB400 0170

This read-only register reflects the current values of corresponding option bits.

Address offset: 0x070

Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>TZEN[7:0]</td>
<td>TrustZone enable configuration bits</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit enables the device is in TrustZone mode during an option byte change.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0xC3: TrustZone disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0xB4: TrustZone enabled.</td>
</tr>
<tr>
<td>23</td>
<td>USBPD_DIS</td>
<td>USB power delivery configuration option bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Disabled</td>
</tr>
<tr>
<td>19</td>
<td>SRAM2_RST</td>
<td>SRAM2 erase when system reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: SRAM2 erased when a system reset occurs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: SRAM2 not erased when a system reset occurs.</td>
</tr>
<tr>
<td>15</td>
<td>SRAM13_RST</td>
<td>SRAM1 and SRAM3 erase upon system reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: SRAM1 and SRAM3 erased when a system reset occurs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: SRAM1 and SRAM3 not erased when a system reset occurs</td>
</tr>
</tbody>
</table>
Bits 1:0  Reserved, must be kept at reset value.

7.11.24  **FLASH option status register 2 (FLASH_OPTSR2_PRG)**

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits. Values after reset reflects the current values of the corresponding option bits.

Address offset: 0x074
Reset value: 0xXXXX XXXX

/Register bits 0 to 31 are loaded with values from the flash memory at OBL/

| Bit | Description                             | Reset Value | Bit 0 | Bit 1 | Bit 2 | Bit 3 | Bit 4 | Bit 5 | Bit 6 | Bit 7 | Bit 8 | Bit 9 | Bit 10 | Bit 11 | Bit 12 | Bit 13 | Bit 14 | Bit 15 |
|-----|-----------------------------------------|-------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|--------|--------|-------|
| 31  | TZEN[7:0]: TrustZone enable configuration bits |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
|     | This bit enables the device is in TrustZone mode during an option byte change. |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
|     | 0xC3: TrustZone disabled                |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
|     | 0xB4: TrustZone enabled                |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
| 23  | Reserved, must be kept at reset value. |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
| 19  | Bit 8 USBPD_DIS: USB power delivery configuration option bit |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
|     | 0: Enabled                             |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
|     | 1: Disabled                            |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
| 18  | Reserved, must be kept at reset value. |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
| 17  | Bit 7 Reserved, must be kept at reset value. |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
| 16  | Bit 6 SRAM2_ECC: SRAM2 ECC detection and correction disable |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
|     | 0: SRAM2 ECC check enabled             |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
|     | 1: SRAM2 ECC check disabled            |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
| 15  | Bit 5 SRAM3_ECC: SRAM3 ECC detection and correction disable |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
|     | 0: SRAM3 ECC check enabled             |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
|     | 1: SRAM3 ECC check disabled            |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
| 14  | Bit 4 BKPRAM_ECC: Backup RAM ECC detection and correction disable |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
|     | 0: BKPRAM ECC check enabled            |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
|     | 1: BKPRAM ECC check disabled            |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
| 13  | Bit 3 SRAM2_RST: SRAM2 erase when system reset |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
|     | 0: SRAM2 erased when a system reset occurs |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |
|     | 1: SRAM2 not erased when a system reset occurs. |             |       |       |       |       |       |       |       |       |       |       |        |        |        |        |        |        |       |

*Note: SRAM erase is triggered by option byte change operation, when enabling this feature.*
Bit 2 **SRAM1_3_RST**: SRAM1 and SRAM3 erase upon system reset
0: SRAM1 and SRAM3 erased when a system reset occurs
1: SRAM1 and SRAM3 not erased when a system reset occurs

*Note*: SRAM erase is triggered by option byte change operation, when enabling this feature.

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

### 7.11.25 FLASH non-secure boot register (FLASH_NSBOOTR_CUR)

This register is non-secure. It can be read by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register reflects the current values of corresponding option bits.

Address offset: 0x080
Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

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

**NSBOOTADD[23:8]**

Bits 31:8 **NSBOOTADD[23:0]**: Non secure unique boot entry address
These bits reflect the Non secure UBE address

Bits 7:0 **NSBOOT_LOCK[7:0]**: A field locking the values of SWAP_BANK, and NSBOOTADD settings.
0xC3: The SWAP_BANK and NSBOOTADD can still be modified following their individual rules.
0xB4: The NSBOOTADD is frozen. SWAP_BANK can only be modified with TZEN set to 0xB4 (enabled).

### 7.11.26 FLASH non-secure boot register (FLASH_NSBOOTR_PRG)

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x084
Reset value: 0xXXXX XXXX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)
A field locking the values of SWAP_BANK, and NSBOOTADD settings.
0xC3: The SWAP_BANK and NSBOOTADD can still be modified following their individual rules.
0xB4: The NSBOOTADD is frozen. SWAP_BANK can only be modified with TZEN set to 0xB4 (enabled).

Bits 31:8  **NSBOOTADD[23:0]**: Non secure unique boot entry address
These bits allow configuring the Non secure BOOT address

Bits 7:0 **NSBOOT_LOCK[7:0]**: A field locking the values of SWAP_BANK, and NSBOOTADD settings.
0xC3: The SWAP_BANK and NSBOOTADD can still be modified following their individual rules.
0xB4: The NSBOOTADD is frozen. SWAP_BANK can only be modified with TZEN set to 0xB4 (enabled).

7.11.27  **FLASH secure boot register (FLASH_SECBOOCTR_CUR)**

This register is secure. It can be read only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

This register reflects the current values of corresponding option bits.

Address offset: 0x088
Reset value: 0xXXXX XXXX
(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

Bits 31:8 **SECBOOTADD[23:0]**: Unique boot entry secure address
These bits reflect the Secure UBE address

Bits 7:0 **SECBOOT_LOCK[7:0]**: A field locking the values of UBE, SWAP_BANK, and SECBOOTADD settings.
0xC3: The BOOT_UBE, SWAP_BANK and SECBOOTADD can still be modified following their individual rules.
0xB4: The BOOT_UBE and SECBOOTADD are frozen. SWAP_BANK can only be modified with TZEN set to 0xC3 (disabled).
7.11.28  FLASH secure boot register (FLASH_BOOTR_PRG)

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x08C

Reset value: 0xXXXX XXXX

(Rегистers bits 0 to 31 are loaded with values from the flash memory at OBL)

| Bits 31:8 | SECBOOTADD[23:0] | Secure unique boot entry address. These bits allow configuring the secure UBE address. |
| Bits 7:0  | SECBOOT_LOCK[7:0] | A field locking the values of UBE, SWAP_BANK, and SECBOOTADD setting. |
|          | 0xC3: The BOOT_UBE, SWAP_BANK and SECBOOTADD can still be modified following their individual rules. |
|          | 0xB4: The BOOT_UBE and SECBOOTADD are frozen. SWAP_BANK can only be modified with TZEN set to 0xC3 (disabled). |

7.11.29  FLASH non-secure OTP block lock (FLASH_OTPBLR_CUR)

This register is non-secure. It can be read by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register reflects the current values of corresponding option bits.

Address offset: 0x090

Reset value: 0xXXXX XXXX
7.11.30FLASH non-secure OTP block lock (FLASH_OTPBLR_PRG)

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x094
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>Bits 31:0 LOCKBL[31:0]: OTP block lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block n corresponds to OTP 16-bit word 32 x n to 32 x n + 31.</td>
</tr>
<tr>
<td>LOCKBL[n] = 1 indicates that all OTP 16-bit words in OTP Block n are locked and attempt to program them results in WRPERR.</td>
</tr>
<tr>
<td>LOCKBL[n] = 0 indicates that all OTP 16-bit words in OTP Block n are not locked.</td>
</tr>
<tr>
<td>When one block is locked, it's not possible to remove the write protection.</td>
</tr>
<tr>
<td>Also if not locked, it is not possible to erase OTP words.</td>
</tr>
</tbody>
</table>

7.11.31FLASH secure block based register for Bank 1 (FLASH_SECBB1Rx)

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI.

Address offset: 0x0A0 + 0x004 * (x-1), (x = 1 to 4)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0 LOCKBL[31:0]: OTP block lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block n corresponds to OTP 16-bit word 32 x n to 32 x n + 31.</td>
</tr>
<tr>
<td>LOCKBL[n] = 1 indicates that all OTP 16-bit words in OTP Block n are locked and attempt to program them results in WRPERR.</td>
</tr>
<tr>
<td>LOCKBL[n] = 0 indicates that all OTP 16-bit words in OTP Block n are not locked.</td>
</tr>
<tr>
<td>When one block is locked, it is not possible to remove the write protection.</td>
</tr>
<tr>
<td>LOCKBL bits can be set if the corresponding bit in FLASH_OTPBLR_CUR is cleared.</td>
</tr>
</tbody>
</table>
7.11.32  FLASH privilege block based register for Bank 1 (FLASH_PRIVBB1Rx)

This register is privileged. It can be read written only by a privileged access. This register can be protected against non-secure write access.

Address offset: 0xC0 + 0x04 * (x-1), (x = 1 to 4)

Reset value: 0x0000 0000

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

Bits 31:0  **PRIVBB1[31:0]**: Privileged/unprivileged 8-Kbyte flash Bank 1 sector attribute (y = 0 to 31)

- 0: sectors (32 * (x-1) + y) in bank 1 is unprivileged.
- 1: sector (32 * (x-1) + y) in bank 1 is privileged.

7.11.33  FLASH security watermark for Bank 1 (FLASH_SECWM1R_CUR)

This register is non-secure. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of corresponding option bits.

Address offset: 0xE0

Reset value: 0x00XX 00XX

(REGISTER bits 0 to 31 are loaded with values from the flash memory at OBL)

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECW M1_END[6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SECW M1_STRT[6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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

Bits 22:16  **SECW M1_END[6:0]**: Bank 1 security WM area 1 end sector

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

Bits 6:0  **SECW M1_STRT[6:0]**: Bank 1 security WM area 1 start sector
7.11.34  **FLASH security watermark for Bank 1 (FLASH_SECWM1R_PRG)**

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x0E4

Reset value: 0x00XX 00XX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

| Bits 31:23 | Reserved, must be kept at reset value. |
| Bits 22:16 | SECWM1_END[6:0]: Bank1 security WM area 1 end sector |
| Bits 15:7  | Reserved, must be kept at reset value. |
| Bits 6:0  | SECWM1_STRT[6:0]: Bank1 security WM area 1 start sector |

7.11.35  **FLASH write sector group protection for Bank 1 (FLASH_WRP1R_CUR)**

This register is non-secure. It can be read by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of corresponding option bits.

Address offset: 0x0E8

Reset value: 0xXXXX XXXX
7.11.36  **FLASH write sector group protection for Bank 1 (FLASH_WRP1R_PRG)**

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x0EC

Reset value: 0xXXXX XXXX

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

Bits 31:0  **WRPG1[31:0]**: Bank 1 sector group protection option status byte

Setting WRPSG1 bits to 0 write protects the corresponding group of four consecutive sectors in bank 1 (0: the group is write-protected; 1: the group is not write-protected)

 Bit 0: Group embedding sectors 0 to 3
 Bit 1: Group embedding sectors 4 to 7
 Bit N: Group embedding sectors 4 x N to 4 x N + 3
 Bit 31: Group embedding sectors 124 to 127

7.11.37  **FLASH data sector configuration Bank 1 (FLASH_EDATA1R_CUR)**

This register is non-secure, can be read by both secure and non-secure access. The register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of corresponding options bits.

Address offset: 0x0F0

Reset value: 0xXXXX XXXX
7.11.38  FLASH data sector configuration Bank 1 (FLASH_EDATA1R_PRG)

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding options bits.

Address offset: 0xF4

Reset value: 0xXXXX XXXX

Bits 31:16  Reserved, must be kept at reset value.

Bit 15  **EDATA1_EN**: Bank 1 flash high-cycle data enable

0: No flash high-cycle data area
1: Flash high-cycle data is used

Bits 14:3  Reserved, must be kept at reset value.

Bits 2:0  **EDATA1_STRT[2:0]**:

EDATA1_STRT contains the start sectors of the flash high-cycle data area in Bank 1. There is no hardware effect to those bits. They shall be managed by ST tools in Flasher.

000: The last sector of the Bank 1 is reserved for flash high-cycle data
001: The two last sectors of the Bank 1 are reserved for flash high-cycle data
010: The three last sectors of the Bank 1 are reserved for flash high-cycle data
...
111: The eight last sectors of the Bank 1 are reserved for flash high-cycle data
7.11.39  FLASH HDP Bank 1 configuration (FLASH_HDP1R_CUR)

This register is non-secure. It can be read by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is initially loaded with the values of corresponding option bits.

Address offset: 0x0F8
Reset value: 0xXXXX XXXX

|   31   |   30   |   29   |   28   |   27   |   26   |   25   |   24   |   23   |   22   |   21   |   20   |   19   |   18   |   17   |   16   |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
|  HDP1_END[6:0] |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        |        |        |        |        |        |  r   |  r   |  r   |  r   |  r   |  r   |  r   |  r   |  r   |  r   |
|   15   |   14   |   13   |   12   |   11   |   10   |   9   |   8   |   7   |   6   |   5   |   4   |   3   |   2   |   1   |   0   |

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

Bits 22:16  HDP1_END[6:0]: HDPL barrier end set in number of 8-Kbyte sectors

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

Bits 6:0  HDP1_STRT[6:0]: HDPL barrier start set in number of 8-Kbyte sectors

7.11.40  FLASH HDP Bank 1 configuration (FLASH_HDP1R_PRG)

This register is non-secure. It can be read and written by both secure and non-secure access, and can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

The register is accessible only in HDPL0 and HDPL1. In HDPL2 or HDPL3 it is WI, RAZ.

This register is used to program values in corresponding option bits.

Address offset: 0x0FC
Reset value: 0xXXXX XXXX
7.11.41 FLASH ECC correction register (FLASH_ECCCORR)

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x100
Reset value: 0x0000 0000

| Bits 31:23 | Reserved, must be kept at reset value. |
| Bits 22:16 | **HDP1_END[6:0]**: HDPL barrier end set in number of 8-Kbyte sectors |
| Bits 15:7  | Reserved, must be kept at reset value. |
| Bits 6:0  | **HDP1_STRT[6:0]**: HDPL barrier start set in number of 8-Kbyte sectors |

### ADDR_ECC[15:0]

| Bits 31   | Reserved, must be kept at reset value. |
| Bit 30   | **ECCC**: ECC correction set by hardware when single ECC error has been detected and corrected. Cleared by writing 1. |
| Bits 29:26 | Reserved, must be kept at reset value. |
| Bit 25   | **ECCCIE**: ECC single correction error interrupt enable bit When ECCCIE bit is set to 1, an interrupt is generated when an ECC single correction error occurs during a read operation. 0: no interrupt generated when an ECC single correction error occurs 1: non-secure interrupt generated when an ECC single correction error occurs |
| Bit 24   | **OTP_ECC**: OTP ECC error bit This bit is set to 1 when a single ECC correction occurred during the last successful read operation from the read-only/OTP area. The address of the ECC error is available in ADDR_ECC bitfield. |
| Bit 23   | **SYSF_ECC**: ECC fail for corrected ECC error in system flash memory It indicates if system flash memory is concerned by ECC error. |
| Bit 22   | **BK_ECC**: ECC fail bank for corrected ECC error It indicates which bank is concerned by ECC error |
Bit 21 **EDATA_ECC**: ECC fail for corrected ECC error in flash high-cycle data area
It indicates if flash high-cycle data area is concerned by ECC error.

Bit 20 **OBK_ECC**: Single ECC error corrected in flash OB Keys storage area. It indicates the OBK storage concerned by ECC error.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDR_ECC[15:0]**: ECC error address
When an ECC error occurs (for single correction) during a read operation, the ADDR_ECC contains the address that generated the error.
ADDR_ECC is reset when the flag error is reset.
The flash interface programs the address in this register only when no ECC error flags are set. This means that only the first address that generated an ECC error is saved.
The address in ADDR_ECC is relative to the flash memory area where the error occurred (user flash memory, system flash memory, data area, read-only/OTP area).

### 7.11.42 FLASH ECC detection register (FLASH_ECCDETR)
This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x104
Reset value: 0x0000 0000
7.11.43  FLASH ECC data (FLASH_ECCDR)

This register is non-secure. It can be read and written by both secure and non-secure
access. This register can be protected against unprivileged access when NSPRIV = 1 in the
FLASH_PRIVCFGR register.

Address offset: 0x108
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 15:0</td>
<td>ADDR_ECC[15:0]: ECC error address</td>
</tr>
<tr>
<td></td>
<td>When an ECC error occurs (double detection) during a read operation, the ADDR_ECC contains the address that generated the error. ADDR_ECC is reset when the flag error is reset. The flash interface programs the address in this register only when no ECC error flags are set. This means that only the first address that generated an double ECC error is saved. The address in ADDR_ECC is relative to the flash memory area where the error occurred (user flash memory, system flash memory, data area, read-only/OTP area).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 15:0</td>
<td>DATA_ECC[15:0]: ECC error data</td>
</tr>
<tr>
<td></td>
<td>When an double detection ECC error occurs on special areas with 6-bit ECC on 16-bit data (data area, read-only/OTP area), the failing data is read to this register. By checking if it is possible to determine whether the failure was on a real data, or due to access to uninitialized memory.</td>
</tr>
</tbody>
</table>

7.11.44  FLASH secure block-based register for Bank 2 (FLASH_SECBB2Rx)

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x1A0 + 0x004 * (x-1), (x = 1 to 4)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 15:0</td>
<td>SECBB2[15:0]: ECC error data</td>
</tr>
<tr>
<td></td>
<td>When an double detection ECC error occurs on special areas with 6-bit ECC on 16-bit data (data area, read-only/OTP area), the failing data is read to this register. By checking if it is possible to determine whether the failure was on a real data, or due to access to uninitialized memory.</td>
</tr>
</tbody>
</table>
Embedded flash memory (FLASH) RM0481

7.11.45 FLASH privilege block-based register for Bank 2 (FLASH_PRIVBB2Rx)

This register is privileged. It can be read written only by a privileged access. This register can be protected against non-secure write access by watermark.

Address offset: 0x1C0 + 0x004 * (x-1), (x = 1 to 4)

Reset value: 0x0000 0000

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

Bits 31:0 PRIVBB2[31:0]: Privileged / non-privileged 8-Kbyte flash Bank 2 sector attribute (y = 0 to 31)

0: sectors (32 * (x-1) + y) in bank 2 is unprivileged
1: sector (32 * (x-1) + y) in bank 2 is privileged

7.11.46 FLASH security watermark for Bank 2 (FLASH_SECWM2R_CUR)

This register is non-secure. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of corresponding option bits.

Address offset: 0x1E0

Reset value: 0x00XX 00XX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

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

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

Bits 22:16 SECWM_END2[6:0]: Bank2 security WM end sector

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

Bits 6:0 SECWM_STRT2[6:0]: Bank2 security WM area start sector
7.11.47  **FLASH security watermark for Bank 2 (FLASH_SECWM2R_PRG)**

This register is secure. It can be read and written only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x1E4
Reset value: 0x00XX 00XX

(Register bits 0 to 31 are loaded with values from the flash memory at OBL)

| Bits 31:23 | Reserved, must be kept at reset value. |
| Bits 22:16 | **SECWM_END2[6:0]**: Bank 2 security WM area end sector |
| Bits 15:7  | Reserved, must be kept at reset value. |
| Bits 6:0   | **SECWM_STRT2[6:0]**: Bank 2 security WM area start sector |

7.11.48  **FLASH write sector group protection for Bank 2 (FLASH_WRP2R_CUR)**

This register is non-secure, can be read by both secure and non-secure access, can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of corresponding option bits.

Address offset: 0x1E8
Reset value: 0xXXXX XXXX

| Bits 31:0  | **WRPSG2[31:0]**: Bank2 sector group protection option status byte |
| Each bit reflects the write protection status of the corresponding group of four consecutive sectors in bank 2 (0: group is write-protected; 1: group is not write-protected) |
| Bit 0: Group embedding sectors 0 to 3 |
| Bit 1: Group embedding sectors 4 to 7 |
| Bit N: Group embedding sectors 4 x N to 4 x N + 3 |
| Bit 31: Group embedding sectors 124 to 127 |
7.11.49  **FLASH write sector group protection for Bank 2**  
(*FLASH_WRP2R_Prg*)

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x1EC

Reset value: 0xXXXX XXXX

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

Bits 31:0  **WRPSG2[31:0]**: Bank 2 sector group protection option status byte

- Setting WRPSGn2 bits to 0 write protects the corresponding group of four consecutive sectors in Bank 2 (0: group is write-protected; 1: group is not write-protected)
- Bit 0: Group embedding sectors 0 to 3
- Bit 1: Group embedding sectors 4 to 7
- Bit N: Group embedding sectors 4 x N to 4 x N + 3
- Bit 31: Group embedding sectors 124 to 127

7.11.50  **FLASH data sectors configuration Bank 2**  
(*FLASH_EDATA2R_CUR*)

This register is non-secure. It can be read by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of corresponding options bits.

Address offset: 0x1F0

Reset value: 0xXXXX XXXX

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

EDATA2_EN: EDATA2_ENABLE

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

EDATA2_STRT[2:0]: EDATA2_START

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

EDATA2_STRT[2:0]: EDATA2_START
Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EDATA2.EN:** Bank 2 flash high-cycle data enable
- 0: No flash high-cycle data area
- 1: Flash high-cycle data is used

Bits 14:3 Reserved, must be kept at reset value.

Bits 2:0 **EDATA2_STRT[2:0]:**
EDATA2_STRT contains the start sectors of the flash high-cycle data area in Bank 2 There is no hardware effect to those bits. They shall be managed by ST tools in Flasher.
- 000: The last sector of the bank 2 is reserved for flash high-cycle data
- 001: The two last sectors of the Bank 2 are reserved for flash high-cycle data
- 010: The three last sectors of the Bank 2 are reserved for flash high-cycle data
...  
**Note:** 111: The eight last sectors of the Bank 2 are reserved for flash high-cycle data.

### 7.11.51 FLASH data sector configuration Bank 2 (FLASH_EDATA2R_PRG)

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding options bits.

Address offset: 0x1F4

Reset value: 0xXXXX XXXX

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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EDATA2.EN:** Bank 2 flash high-cycle data enable
- 0: No flash high-cycle data area
- 1: Flash high-cycle data is used

Bits 14:3 Reserved, must be kept at reset value.

Bits 2:0 **EDATA2_STRT[2:0]:**
EDATA2_STRT contains the start sectors of the flash high-cycle data area in Bank 2 There is no hardware effect to those bits. They shall be managed by ST tools in Flasher.
- 000: The last sector of the Bank 2 is reserved for flash high-cycle data
- 001: The two last sectors of the Bank 2 are reserved for flash high-cycle data
- 010: The three last sectors of the Bank 2 are reserved for flash high-cycle data
...  
**Note:** 111: The eight last sectors of the Bank 2 are reserved for flash high-cycle data.
7.11.52  FLASH HDP Bank 2 configuration (FLASH_HDP2R_CUR)

This register is initially loaded with the values of corresponding option bits. This register is non-secure. It can be read by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x1F8
Reset value: 0xXXXX XXXX

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

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

Bits 22:16  **HDP2_END[6:0]**: HDPL barrier end set in number of 8-Kbyte sectors

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

Bits 6:0  **HDP2_STRT[6:0]**: HDPL barrier start set in number of 8-Kbyte sectors

7.11.53  FLASH HDP Bank 2 configuration (FLASH_HDP2R_PRG)

This register is non-secure. It can be read and written only by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register. Register is accessible only in HDPL0 and HDPL1. In HDPL2 or HDPL3 it is WI, RAZ.

This register is used to program values in corresponding option bits.

Address offset: 0x1FC
Reset value: 0xXXXX XXXX

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

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

Bits 22:16  **HDP2_END[6:0]**: HDPL barrier end set in number of 8-Kbyte sectors

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

Bits 6:0  **HDP2_STRT[6:0]**: HDPL barrier start set in number of 8-Kbyte sectors
## 7.12 FLASH register map and reset values

### Table 75. Register map and reset value table

|--------|---------------|-------------|------|--------|--------|----------|---------------|---------|--------|---------------------|
### Table 75. Register map and reset value table (continued)

| Offset | Register name | reset value | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x02C  | FLASH_SECCR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | B.KSEL       | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Res          | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Reset value  | 0 0          | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x030  | FLASH_NSCCR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Reset value  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x034  | FLASH_SECCCR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Reset value  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x03C  | FLASH_PRIVCFGR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Reset value  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x040  | FLASH_NSOBKCFGR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | SWAP_OFFSET[8:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Reset value  | 1 1 1 1 1 1 1 1 1 1 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x044  | FLASH_SECOBKCFGR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | SWAP_OFFSET[8:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Reset value  | 1 1 1 1 1 1 1 1 1 1 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x048  | FLASH_HDPEXTR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | HDP2_EXT[6:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | HDP1_EXT[6:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Reset value  | 0 0 0 0 0 0 0 0 0 0 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x04C  | FLASH_OPTSR_CUR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x050  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|        | Reset value  | X X X X X X X X X X X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Note: The table continues with more registers and their reset values. The last row shows the reset values for registers starting from 0x04C to 0x050.
### Table 75. Register map and reset value table (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x054</td>
<td>FLASH_OPTSR_PRG</td>
<td><img src="image.png" alt="Register Map" /></td>
</tr>
<tr>
<td>0x058</td>
<td>Reserved</td>
<td><img src="image.png" alt="Table Entry" /></td>
</tr>
<tr>
<td>0x05C</td>
<td>Reserved</td>
<td><img src="image.png" alt="Table Entry" /></td>
</tr>
<tr>
<td>0x060</td>
<td>FLASH_NSEPOCHR_CUR</td>
<td><img src="image.png" alt="Table Entry" /></td>
</tr>
<tr>
<td>0x064</td>
<td>FLASH_NSEPOCHR_PRG</td>
<td><img src="image.png" alt="Table Entry" /></td>
</tr>
<tr>
<td>0x068</td>
<td>FLASH_SECEPOCHR_CUR</td>
<td><img src="image.png" alt="Table Entry" /></td>
</tr>
<tr>
<td>0x06C</td>
<td>FLASH_SECEPOCHR_PRG</td>
<td><img src="image.png" alt="Table Entry" /></td>
</tr>
<tr>
<td>0x070</td>
<td>FLASH_OPTSR2_CUR</td>
<td><img src="image.png" alt="Table Entry" /></td>
</tr>
<tr>
<td>0x074</td>
<td>FLASH_OPTSR2_PRG</td>
<td><img src="image.png" alt="Table Entry" /></td>
</tr>
<tr>
<td>0x080</td>
<td>FLASH_NSBOOTR_CUR</td>
<td><img src="image.png" alt="Table Entry" /></td>
</tr>
<tr>
<td>0x084</td>
<td>FLASH_NSBOOTR_PRG</td>
<td><img src="image.png" alt="Table Entry" /></td>
</tr>
<tr>
<td>0x088</td>
<td>FLASH_SECBOOTR_CUR</td>
<td><img src="image.png" alt="Table Entry" /></td>
</tr>
<tr>
<td>0x08C</td>
<td>FLASH_SECBOOTR_PRG</td>
<td><img src="image.png" alt="Table Entry" /></td>
</tr>
<tr>
<td>0x090</td>
<td>FLASH_OTPBLR_CUR</td>
<td><img src="image.png" alt="Table Entry" /></td>
</tr>
<tr>
<td>0x094</td>
<td>FLASH_OTPBLR_PRG</td>
<td><img src="image.png" alt="Table Entry" /></td>
</tr>
</tbody>
</table>
Table 75. Register map and reset value table (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name _CUR1R</th>
<th>Sec.</th>
<th>Offset</th>
<th>Register name _PRG1R</th>
<th>Sec.</th>
<th>Offset</th>
<th>Register name CUR1R</th>
<th>Sec.</th>
<th>Offset</th>
<th>Register name PRG1R</th>
<th>Sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0A0 + 0x004 * x (x = 1 to 4)</td>
<td>FLASH_SECBB1R_x</td>
<td>SECBB1[31:0]</td>
<td>0x0C0 + 0x004 * x (x = 1 to 4)</td>
<td>FLASH_PRIVBB1R_x</td>
<td>PRIVBB1[31:0]</td>
<td>0x0D0 - 0x0DC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0E0</td>
<td>FLASH_SECWM_SECWM1_END[6:0]</td>
<td>SECWM1_END[6:0]</td>
<td>0x0E4</td>
<td>FLASH_SECWM_SECWM1_END[6:0]</td>
<td>SECWM1_END[6:0]</td>
<td>0x0E8</td>
<td>FLASH_WRP_WRP1_END[6:0]</td>
<td>WRPSG1[31:0]</td>
<td>0x0F0</td>
<td>FLASH_EDATA_EDATA1_END[2:0]</td>
<td>EDATA1_STRT[2:0]</td>
</tr>
<tr>
<td>0x0F0</td>
<td>FLASH_EDATA_SECWM1_END[6:0]</td>
<td>SECWM1_END[6:0]</td>
<td>0x0F4</td>
<td>FLASH_EDATA_SECWM1_END[6:0]</td>
<td>SECWM1_END[6:0]</td>
<td>0x0FC</td>
<td>FLASH_EDATA_SECWM1_END[6:0]</td>
<td>SECWM1_END[6:0]</td>
<td>0x100</td>
<td>FLASH_EDATA_ADDR_ECC[15:0]</td>
<td></td>
</tr>
</tbody>
</table>

Reset value

| 0x0 | 0x1 | 0x2 | 0x3 | 0x4 | 0x5 | 0x6 | 0x7 | 0x8 | 0x9 | 0x10 | 0x11 | 0x12 | 0x13 | 0x14 | 0x15 | 0x16 | 0x17 | 0x18 | 0x19 | 0x1A | 0x1B | 0x1C | 0x1D | 0x1E | 0x1F | 0x20 | 0x21 | 0x22 | 0x23 | 0x24 | 0x25 | 0x26 | 0x27 | 0x28 | 0x29 | 0x2A | 0x2B | 0x2C | 0x2D | 0x2E | 0x2F |
Table 75. Register map and reset value table (continued)

| Offset | Register name | Reset value | offset |
|--------|---------------|-------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|#####
Refer to Section 2.3 for the register boundary addresses.
8 Instruction cache (ICACHE)

8.1 ICACHE introduction

The instruction cache (ICACHE) is introduced on C-AHB code bus of Cortex-M33 processor to improve performance when fetching instruction and data from internal and external memories.

Some specific features like dual master ports, hit-under-miss and critical-word-first refill policy, allow close to zero-wait-state performance in most use cases.

8.2 ICACHE main features

The main features of ICACHE are described below:

- **Bus interface**
  - one 32-bit AHB slave port, the execution port (input from Cortex-M33 C-AHB code interface)
  - two AHB master ports: master1 and master2 ports (outputs to Fast and Slow buses of main AHB bus matrix, respectively)
  - one 32-bit AHB slave port for control (input from AHB peripherals interconnect, for ICACHE registers access)

- **Cache access**
  - 0 wait-state on hits
  - Hit-under-miss capability: ability to serve processor requests (access to cached data) during an ongoing line refill due to a previous cache miss
  - Dual master access: feature used to decouple the traffic according to targeted memory. For example, the ICACHE assigns fast traffic (addressing flash and SRAM memories) to the AHB master1 port, and slow traffic (addressing external memories) to AHB master2 port, thus preventing processor stalls on lines refills from external memories. This allows ISR (interrupt service routine) fetching on internal flash memory to take place in parallel with a cache line refill from external memories.
  - Minimal impact on interrupt latency, thanks to dual master
  - Optimal cache line refill thanks to WRAPw bursts of the size of the cache line (32-bit word size, w, aligned on cache line size)
  - n-way set-associative default configuration with possibility to configure as 1-way, means direct mapped cache, for applications needing very-low-power consumption profile

- **Memory address remap**
  - Possibility to remap input address falling into up to four memory regions (used to remap aliased code in external memories to the Code region, for execution from C-AHB code interface)

- **Replacement and refill**
  - pLRU-t replacement policy (pseudo-least-recently-used, based on binary tree), algorithm with best complexity/performance balance
  - Critical-word-first refill policy, minimizing processor stalls
- Possibility to configure burst type of AHB memory transaction for remapped regions: INCRw or WRAPw (size w aligned on cache line size)

**Performance counters**
- The ICACHE implements two performance counters:
  - Hit monitor counter (32-bit)
  - Miss monitor counter (16-bit)

**Error management**
- Possibility to detect an unexpected cacheable write access, to flag an error and optionally to raise an interrupt

**TrustZone security support**

**Maintenance operation**
- Cache invalidate: full cache invalidation, fast command, non interruptible

### 8.3 ICACHE implementation

**Table 76. ICACHE features**

<table>
<thead>
<tr>
<th>Feature</th>
<th>ICACHE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of ways</td>
<td>2</td>
</tr>
<tr>
<td>Cache size</td>
<td>8 Kbytes</td>
</tr>
<tr>
<td>Cache line width</td>
<td>16 bytes</td>
</tr>
<tr>
<td>Range granularity of memory regions to be remapped</td>
<td>2 Mbytes</td>
</tr>
<tr>
<td>Number of regions to remap</td>
<td>4</td>
</tr>
<tr>
<td>Data size of AHB slave interface</td>
<td>32 bits</td>
</tr>
<tr>
<td>Data size of AHB fast master1 interface</td>
<td>128 bits</td>
</tr>
<tr>
<td>Data size of AHB slow master2 interface</td>
<td>32 bits</td>
</tr>
</tbody>
</table>

### 8.4 ICACHE functional description

The purpose of the instruction cache is to cache instruction fetches or instruction memories loads, coming from the processor. As such, the ICACHE only manages read transactions and does not manage write transactions.

For error management purpose, in case a write cacheable transaction is presented (this only happens in case of bad software programming), the ICACHE sets an error flag and, if enabled, raises an interrupt to the processor.
8.4.1 ICACHE block diagram

Figure 31. ICACHE block diagram

8.4.2 ICACHE reset and clocks

The ICACHE is clocked on Cortex-M33 C-AHB bus clock.

When the ICACHE reset signal is released, a cache invalidate procedure is automatically launched, making the ICACHE busy (ICACHE_SR = 0x0000 0001).

When this procedure is finished:

- the ICACHE is invalidated: “cold cache”, with all cache line valid bits = 0 (ICACHE must be filled up)
- ICACHE_SR = 0x0000 0002 (reflecting the cache is no more busy)
- the ICACHE is disabled: the EN bit in ICACHE_CR holds its reset state (=0).

*Note:* When disabled, the ICACHE is bypassed, except the remapping mechanism that is still functional: slave input requests (remapped or not) are just forwarded to the master port(s).
### 8.4.3 ICACHE TAG memory

The ICACHE TAG memory contains:
- address tags, that indicate which data are contained in the cache data memories
- validity bits

There is one valid bit per cache line (per way).

The valid bit is set when a cache line is refilled (after a miss).

Valid bits are reset in any of the below cases:
- after the ICACHE reset is released
- when the cache is disabled, by setting EN = 0 in ICACHE_CR (by software)
- when executing an ICACHE invalidate command, by setting CACHEINV = 1 in ICACHE_CR (by software)

When a cacheable transaction is received at the execution input port, its AHB address (HADDR_in) is split into the following fields (see Table 77 for B and W definitions):
- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to the TAG memory address to check if the requested data is already available (meaning valid) inside the ICACHE.

The table below gives a summary of ICACHE main parameters for TAG memory dimensioning. Figure 32 shows the functional view of TAG and data memories, for an n-way set associative ICACHE.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache size</td>
<td>S Kbytes = s bytes (s = 1024 x S)</td>
<td>8 Kbytes = 8192 bytes</td>
</tr>
<tr>
<td>Cache number of ways</td>
<td>n</td>
<td>2</td>
</tr>
<tr>
<td>Cache line size</td>
<td>L-byte = l-bit (l = 8 x L)</td>
<td>16-byte = 128-bit</td>
</tr>
<tr>
<td>Number of cache lines (per way)</td>
<td>LpW = s / (n x L) lines / way</td>
<td>256 lines / way</td>
</tr>
<tr>
<td>Address byte offset size</td>
<td>B = log2(L) bit</td>
<td>4-bit</td>
</tr>
<tr>
<td>Address way index size</td>
<td>W = log2(LpW) bit</td>
<td>8-bit</td>
</tr>
<tr>
<td>TAG address size</td>
<td>T = (32 - W - B) bit</td>
<td>20-bit</td>
</tr>
</tbody>
</table>
8.4.4  Direct-mapped ICACHE (1-way cache)

The default configuration (at reset) is an n-way set associative cache (WAYSEL = 1 in ICACHE_CR), but the user can configure the ICACHE as direct mapped by writing WAYSEL = 0 (only possible when the cache is disabled, EN = 0 in ICACHE_CR).

The table below gives a summary of ICACHE main parameters for TAG memory when the direct-mapped cache operating mode is selected.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache size</td>
<td>S Kbytes = s bytes (s = 1024 x S)</td>
<td>8 Kbytes = 8192 bytes</td>
</tr>
<tr>
<td>Cache number of ways</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Cache line size</td>
<td>L-byte = l-bit (l = 8 x L)</td>
<td>16-byte = 128-bit</td>
</tr>
<tr>
<td>Number of cache lines</td>
<td>LpW = s / L lines</td>
<td>512 lines</td>
</tr>
<tr>
<td>Address byte offset size</td>
<td>B = log2(L) bit</td>
<td>4-bit</td>
</tr>
<tr>
<td>Address way index size</td>
<td>W = log2(LpW) bit</td>
<td>9-bit</td>
</tr>
<tr>
<td>TAG address size</td>
<td>T = (32 - W - B) bit</td>
<td>19-bit</td>
</tr>
</tbody>
</table>
All cache operations (such as read, refill, remapping, invalidation) remain the same in direct-mapped configuration. The only difference is the absence of a replacement algorithm in case of line eviction (as explained in Section 8.4.8): only one way (the unique one) is possible for any data refill.

8.4.5 ICACHE enable

To activate the ICACHE, the EN bit must be set to 1 in ICACHE_CR.

When the ICACHE is disabled, it is bypassed and all transactions are copied from the slave port to the master ports in the same clock cycle.

It is recommended to initialize or modify the main memory content (region to be later cached) with the ICACHE disabled, and to enable the ICACHE only when this region remains unchanged (an enabled ICACHE detects cacheable write transactions as errors).

In order to insure performance determinism, it is recommended to wait for the end of a potential cache invalidate procedure before enabling the ICACHE. This invalidate procedure occurs when the hardware reset signal is released, when CACHEINV is set, or when EN is cleared in ICACHE_CR. During the procedure, BUSYF is set in ICACHE_SR, and once finished, BUSYF is cleared and BSYENDF is set in the same register (raising the ICACHE interrupt if enabled on such a busy end condition).

The software must test BUSYF and/or BSYENDF values before enabling the ICACHE. Else, if the ICACHE is enabled before the end of an invalidate procedure, any cache access (while BUSYF = 1) is treated as non cacheable, and its performance depends on the main memory access time.

The address remapping is performed, whether the ICACHE is enabled or not, if the input transaction address falls into memory regions defined and enabled in ICACHE_CRRx (see Figure 33).

The ICACHE is by default disabled at boot.

8.4.6 Cacheable and non-cacheable traffic

The ICACHE is developed for Cortex-M33 core. It is placed on C-AHB bus, and thus caches the code memory region, ranging from address 0x0000 0000 to 0xFFFF FFFF of the memory map.

In order to make some other memory regions cacheable, the ICACHE supports a memory-region-remapping feature. It is used to define up to four external memory regions, which addresses have an alias in the code region. Addressing these external memory regions through their code alias address allows the memory request to be routed to the C-AHB bus, and to be managed by the ICACHE.

Any external memory space physically mapped at an address in range [0x6000 0000:0xAFFF FFFF] can be aliased with an address in range [0x0000 0000:0x07FF FFFF] or [0x1000 0000:0x1FFF FFFF].

For a given memory request in the code region, the ICACHE implements the address remapping functionality first. If aliased, it is the remapped address which is then cached, and, if needed, provided to the master port to address the main AHB bus matrix. The destination physical address does not need further manipulation on the AHB bus.

The remapping functionality is available also for non-cacheable traffic, and when the cache is disabled.
Further details on address remapping are provided in Section 8.4.7.

An incoming memory request to the ICACHE is defined as cacheable according to its AHB transaction memory lookup attribute, as shown in Table 79. This AHB attribute depends on the MPU (memory protection unit) programming for the addressed region.

Table 79. ICACHE cacheability for AHB transaction

<table>
<thead>
<tr>
<th>AHB lookup attribute</th>
<th>Cacheability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cacheable</td>
</tr>
<tr>
<td>0</td>
<td>Non cacheable</td>
</tr>
</tbody>
</table>

In case of a non-cacheable access, the ICACHE is bypassed, meaning that the AHB transaction is propagated unchanged to the master output port, except the transaction address which may be modified due to the address remapping feature (see Section 8.4.7).

The bypass, and eventual remap logic, does not increase the latency of the access to the targeted memory.

In case of a cacheable access, the ICACHE behaves as explained in Section 8.4.8.

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

The table below summarizes programmable configurations of various memories.

Table 80. Memory configurations

<table>
<thead>
<tr>
<th>Memory</th>
<th>Cacheable (MPU programming)</th>
<th>Remapped in the ICACHE (ICACHE_CRRx programming)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flash memory</td>
<td>Yes or No</td>
<td>Not required</td>
</tr>
<tr>
<td>SRAM</td>
<td>Not recommended</td>
<td>Not required</td>
</tr>
<tr>
<td>External memories</td>
<td>Yes</td>
<td>Required</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>Required if the user wants code in external memories fetched on C-AHB bus (else on S-AHB bus)</td>
</tr>
</tbody>
</table>

8.4.7 Address remapping

The ICACHE allows an alias address to be defined in the code region for up to four external memory regions.

The address remapping is applied on the code alias address, transforming it into the destination external physical address.

The remapping operation is fully software configurable by programming ICACHE_CRRx (x = 0 to 3, number of remapped regions). This programming can be done only when the ICACHE is disabled.

Each region x can be individually enabled with REN in ICACHE_CRRx. Once enabled, the remap operation occurs even if the ICACHE is disabled, or if the transaction is not cacheable.
Remap regions can have different size: each region size can be programmed in RSIZE of its ICACHE_CRRx. The size of each region is a power of two multiple of range granularity (2 Mbytes), with a minimum region size of 2 Mbytes, and a maximum region size of 128 Mbytes.

The address remapping mechanism is based on the matching of an incoming AHB address (HADDR_in) with a given code subregion base-address, and the modification of this address into its (remapped) external physical address, as follows:

- **HADDR_in** belongs to region x if HADDR_in[31:RI] = 000:BASEADDR[28:RI], where:
  - 000:BASEADDR is the code subregion base-address programmed in BASEADDR of ICACHE_CRRx.
  - RI defines the number of significant bits to consider. RI = log2(region size) with a minimum value of 21 (for a 2-Mbyte region) and a maximum value of 27 (for a 128-Mbyte region).
- If the region x is enabled, the master port output AHB address (HADDR_out) is composed by concatenating the two below parts:
  - REMAPADDR[31:RI] field of ICACHE_CRRx as MSBs
  - HADDR_in[RI-1:0] as LSBs.

The figure below describes the matching and the output address generation.

![Figure 33. ICACHE remapping address mechanism](image)

The table below summarizes all possible configurations of BASEADDR and REMAPADDR sizes (number of significant MSBs) in ICACHE_CRRx, depending on RSIZE.

<table>
<thead>
<tr>
<th>Region size (Mbytes)</th>
<th>Base address size (MSBs)</th>
<th>Remap address (MSBs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>64</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>128</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>
Care must be taken while programming BASEADDR and REMAPADDR in ICACHE_CRRx: if the programmed value is bigger than expected (number of MSBs, see Table 81), the unnecessary extra LSBs are ignored.

Typical remapping example: a 128-Mbyte FMC region (NOR/SRAM) physically located in the external address range [0x6800 0000:0x6FFF FFFF], remapped in the code section range [0x1000 0000:0x17FF FFFF]:

- REMAPADDR[31:21] = 0x340
- BASEADDR[28:21] = 0x80
- HADDR_in[31:27] is compared to 000:BASEADDR[28:27], and HADDR_in/BASEADDR[26:21] are ignored for the comparison.

If the comparison matches:

- HADDR_out[31:27] gets REMAPADDR[31:27] (in place of HADDR_in[31:27])
- HADDR_out[26:0] gets HADDR_in[26:0]

The software can program the kind of AHB burst that is generated by the ICACHE master ports on the bus matrix (for cache line refill), by setting HBURST in ICACHE_CRRx with:

- WRAP for remapped external memories accessed through interfaces that can support WRAP burst mode, providing the benefit of the critical-word-first feature performance:
  - WRAP burst size = cache line size
  - WRAP burst start address = word address of the first data requested by the core
- INCR: INCR burst mode for external memories accessed through the interfaces that do not support WRAP burst mode (losing the benefit of critical-word-first feature):
  - INCR burst size = cache line size
  - INCR burst start address = address aligned on the boundary of the cache line containing the requested word.

Note: Coherency is needed when programming the SAU (secure attribution unit) and the MPU (memory protection unit) attributes for both the external regions and their aliased code subregions.

8.4.8 Cacheable accesses

When the ICACHE receives a cacheable transaction from the Cortex-M33, the ICACHE checks if the address requested is present in its TAG memory, and if the corresponding cache line is valid.

There are then three alternatives:

- The address is present inside the TAG memory, the cache line is valid: cache hit, the data is read from the cache and provided to the processor in the same cycle.
- The address is not present in the TAG memory: cache miss, the data is read from the main memory and provided to the processor, and a cache line refill is performed. The critical-word-first policy insures minimum wait cycles for the processor, since read data can be provided while the cache still performs a cache line refill (associated latency is the latency of fetching one word from the main memory).
  
The burst generated on the ICACHE master bus is WRAPw (w being the cache line width, in words) in case no address remap occurs. If an address remap occurs, the kind of burst depends on HBURST programmed in corresponding ICACHE_CRRx.

The AHB transaction attributes are also propagated to the main AHB bus matrix on the master port selected for the line refill.
• The address is not present in TAG memory, but belongs to the refill burst from the main memory that is currently ongoing: cache hit (hit-under-miss feature).

This happens during cache-line refill. The ICACHE can provide the requested data as soon as the data is available at its master interface, thus avoiding a miss (fetching data from the main memory).

In case of cache refill (due to cache miss), the ICACHE selects which cache line is written with the refill data:
• In direct map (1-way) mode, only one line can be used to store the refill data: the line pointed by the index of the input address.
• In n-way set associative mode, one line among n can be used (the line pointed by the address index, in each of the n ways). The way selection is based on a pLRU-t replacement algorithm, that points, for each index, on the way candidate for the next refill.

If ever the cache line where the refill data must be written is already valid, the targeted cache line must be invalidated first. This is true whatever the direct map or n-way set associative cache mode.

8.4.9 Dual-master cache

The ICACHE can implement a dual-port AHB master on the main AHB bus matrix: master1 and master2 ports. This is used to split the traffic going to different destination memories.

The non-remapped traffic goes systematically to master1 port. The re-mapped traffic to external memories must be routed on master2 port by programming MSTSEL in ICACHE_CRRx (on a region basis).

The code can typically be fetched as follows:
• internal flash memory and internal SRAM on master1 port (Fast bus)
• external flash memory/RAM on master2 port (Slow bus)

For systems not implementing external memories, the traffic to the internal flash memory can be decoupled from the traffic to the internal SRAM (when remapped by the ICACHE). This feature is used to prevent further processor stalls on misses.

Alongside with hit-under-miss, this dual-master feature allows the processor to have an alternative path in case of fetching from different memories.

8.4.10 ICACHE security

The ICACHE implements an Armv8-M TrustZone.

ICACHE configuration registers are protected at system level.

8.4.11 ICACHE maintenance

The software can invalidate the whole content of the ICACHE by programming CACHEINV in ICACHE_CR register.

When CACHEINV = 1, the ICACHE control logic sets BUSYF flag in ICACHE_SR and launches the invalidate cache operation, Resetting each TAG valid bit to 0 (one valid bit per cache line). CACHEINV is automatically cleared.

Once the invalidate operation is finished, the ICACHE automatically clears BUSYF, and sets BSYENDF in ICACHE_SR register.
If enabled on this flag condition (BSYENDIE = 1 in ICACHE_IER), the ICACHE interrupt is raised. Then, the (empty) cache is available again.

### 8.4.12 ICACHE performance monitoring

The ICACHE provides the following monitors for performance analysis:

- The 32-bit hit monitor counts the cacheable AHB-transactions on the slave cache port that hits the ICACHE content.
  
  It also takes into account all accesses whose address is present in the TAG memory or in the refill buffer (due to a previous miss, and whose data is coming, or is soon to come, from cache master port) (see Section 8.4.8).

- The 16-bit miss monitor counts the cacheable AHB-transactions on the slave cache port that misses the ICACHE content.
  
  It also takes into account all accesses whose address is not present neither in the TAG memory nor in the refill buffer.

Upon reaching their maximum values, these monitors do not wrap over. Hit and miss monitors can be enabled and reset by software allowing the analysis of specific pieces of code.

The software can perform the following tasks:

- Enable/stop the hit monitor through HITMEN in ICACHE_CR.
- Reset the hit monitor by setting HITMRST in ICACHE_CR.
- Enable/stop the miss monitor through MISSMEN in ICACHE_CR.
- Reset the miss monitor by setting MISSMRST in ICACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

### 8.4.13 ICACHE boot

The ICACHE is disabled (EN = 0 in ICACHE_CR) at boot.

The code remapping at boot is not needed for Cortex-M33 since it implements the VTOR (vector tables) that allows a boot start address definition different than 0x0.

Once the boot is finished, the ICACHE can be enabled (software setting EN = 1 in CACHE_CR).

### 8.5 ICACHE low-power modes

At device level, using the ICACHE reduces the power consumption by fetching instructions from the internal ICACHE most of the time, rather than from the bigger and then more-power-consuming main memories. This reduction is even higher if the cached main memories are external.

Applications with a lower-performance profile (in terms of hit ratio) and stringent low-power consumption constraints may benefit from the lower power consumption of an ICACHE configured as direct mapped. This single-way cache configuration is obtained by programming WAYSEL = 0 in ICACHE_CR (see Figure 32). The power consumption is then reduced by accessing, for each request, only the necessary cut of TAG and data memories. Meanwhile, the cache effect still improves fetch performance. Even if for most code execution, it is a little less efficient than with an n-way set associative cache mode.
8.6 ICACHE error management and interrupts

In case an unsupported cacheable write request is detected (functional error), the ICACHE generates an error by setting the ERRF flag in ICACHE_SR. An interrupt is then generated if the corresponding interrupt enable bit is set (ERRIE = 1 in ICACHE_IER).

The other possible interrupt generation is at the end of a cache invalidation operation. When the cache-busy state is finished, the ICACHE sets BSYENDF flag in ICACHE_SR. An interrupt is then generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in ICACHE_IER).

All ICACHE interrupt sources raise the same and unique interrupt signal, icache_it, and then use the same interrupt vector.

<table>
<thead>
<tr>
<th>Table 82. ICACHE interrupts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt vector</td>
</tr>
<tr>
<td>ICACHE</td>
</tr>
<tr>
<td>End of busy state (invalidate finished)</td>
</tr>
</tbody>
</table>

The ICACHE also propagates all AHB bus errors (such as security issues, address decoding issues) from master1 or master2 port back to the execution port.

8.7 ICACHE registers

8.7.1 ICACHE control register (ICACHE_CR)

Address offset: 0x000
Reset value: 0x0000 0004

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

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 MISSMRST: miss monitor reset
0: no effect
1: reset cache miss monitor

Bit 18 HITMRST: hit monitor reset
0: no effect
1: reset cache hit monitor
Bit 17 **MISSMEN**: miss monitor enable
0: cache miss monitor switched off. Stopping the monitor does not reset it.
1: cache miss monitor enabled

Bit 16 **HITMEN**: hit monitor enable
0: cache hit monitor switched off. Stopping the monitor does not reset it.
1: cache hit monitor enabled

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **WAYSEL**: cache associativity mode selection
This bit allows user to choose ICACHE set-associativity. It can be written by software only when cache is disabled (EN = 0).
0: direct mapped cache (1-way cache)
1: n-way set associative cache (reset value)

Bit 1 **CACHEINV**: cache invalidation
Set by software and cleared by hardware when the BUSYF flag is set (during cache maintenance operation). Writing 0 has no effect.
0: no effect
1: invalidate entire cache (all cache lines valid bit = 0)

Bit 0 **EN**: enable
0: cache disabled
1: cache enabled

### 8.7.2 ICACHE status register (ICACHE_SR)

Address offset: 0x004
Reset value: 0x0000 0001

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

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

Bit 2 **ERRF**: cache error flag
0: no error
1: an error occurred during the operation (cacheable write)

Bit 1 **BSYEN DF**: busy end flag
0: cache busy
1: full invalidate CACHEINV operation finished

Bit 0 **BUSYF**: busy flag
0: cache not busy on a CACHEINV operation
1: cache executing a full invalidate CACHEINV operation
8.7.3 ICACHE interrupt enable register (ICACHE_IER)

Address offset: 0x008
Reset value: 0x0000 0000

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

Bit 2 **ERRIE**: interrupt enable on cache error
Set by software to enable an interrupt generation in case of cache functional error (cacheable write access).
0: interrupt disabled on error
1: interrupt enabled on error

Bit 1 **BSYENDIE**: interrupt enable on busy end
Set by software to enable an interrupt generation at the end of a cache invalidate operation.
0: interrupt disabled on busy end
1: interrupt enabled on busy end

Bit 0 Reserved, must be kept at reset value.

8.7.4 ICACHE flag clear register (ICACHE_FCR)

Address offset: 0x00C
Reset value: 0x0000 0000

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

Bit 2 **CERRF**: clear cache error flag
Set by software.
0: no effect
1: clears ERF flag in ICACHE_SR

Bit 1 **CBSYENDF**: clear busy end flag
Set by software.
0: no effect
1: clears BSYENDF flag in ICACHE_SR.
8.7.5 **ICACHE hit monitor register (ICACHE_HMONR)**

Address offset: 0x010  
Reset value: 0x0000 0000

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

Bits 31:0 **HITMON[31:0]**: cache hit monitor counter

8.7.6 **ICACHE miss monitor register (ICACHE_MMONR)**

Address offset: 0x014  
Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.  
Bits 15:0 **MISSMON[15:0]**: cache miss monitor counter

8.7.7 **ICACHE region x configuration register (ICACHE_CRRx)**

Address offset: 0x020 + 0x4 * x, (x = 0 to 3)  
Reset value: 0x0000 0200

Define an alias address in Code region for other regions, making them cacheable.  
BASEADDR and REMAPADDR fields are write locked (read only) when EN = 1 in ICACHE_CR.
Bit 31 **HBURST**: output burst type for region x
0: WRAP
1: INCR

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **MSTSEL**: AHB cache master selection for region x
0: no action (master1 selected by default)
1: master2 selected

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **REMAPADDR[31:21]**: remapped address for region x
This field replaces the alias address defined by BASEADDR field.
The only useful bits are [31:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see Section 8.4.7). If the programmed value has more LSBs, the useless bits are ignored.

Bit 15 **REN**: enable for region x
0: disabled
1: enabled

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:9 **RSIZE[2:0]**: size for region x
000: reserved
001: 2 Mbytes
010: 4 Mbytes
011: 8 Mbytes
100: 16 Mbytes
101: 32 Mbytes
110: 64 Mbytes
111: 128 Mbytes

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **BASEADDR[28:21]**: base address for region x
This alias address is replaced by REMAPADDR field.
The only useful bits are [28:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see Section 8.4.7). If the programmed value has more LSBs, the useless bits are ignored.

### 8.7.8 ICACHE register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>0x000</th>
<th>0x004</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ICACHE_CR</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ICACHE_SR</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 83. ICACHE register map and reset values
### Table 83. ICACHE register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x08</td>
<td>ICACHE_IER</td>
<td>0x0C</td>
<td>ICACHE_FCR</td>
<td>0x10</td>
<td>ICACHE_HMONR</td>
<td>0x14</td>
<td>ICACHE_MMONR</td>
<td>0x18-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1C</td>
<td></td>
<td></td>
<td></td>
<td>0x20</td>
<td>ICACHE_CRR0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x24</td>
<td>ICACHE_CRR1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x28</td>
<td>ICACHE_CRR2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x2C</td>
<td>ICACHE_CRR3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for the register boundary addresses.
9 Data cache (DCACHE)

9.1 DCACHE introduction

The data cache (DCACHE) is introduced on S-AHB system bus of Cortex-M33 processor to improve the performance of data traffic to/from external memories.

Some specific features like hit-under-miss and critical-word-first refill policy allow optimum performance on external memories data accesses.

9.2 DCACHE main features

The main features of DCACHE are described below:

- **Bus interface**
  - one 32-bit AHB slave port, the system port (input from Cortex-M33 S-AHB system interface)
  - one 32-bit AHB master port (output to main AHB bus matrix)
  - one 32-bit AHB slave port for control (input from AHB peripherals interconnect, for access to DCACHE registers)

- **Cache access**
  - 0 wait-state on hits
  - Hit-under-miss capability: ability to serve processor requests (access to cached data) during an ongoing line refill due to a previous cache miss
  - Optimized cache line refill thanks to WRAP bursts of the size of the cache line (such as WRAP4 for 128-bit cache line)
  - 2-ways set-associative
  - Supports both write-back and write-through policies (selectable with AHB bufferable attribute)
  - Read and write-back always allocate
  - Write-through always non-allocate (write-around)
  - Supports byte, half-word and word writes

- **Replacement and refill**
  - pLRU-t replacement policy (pseudo-least-recently-used, based on binary tree), algorithm with best complexity/performance balance
  - Critical-word-first refill policy for read transactions, minimizing processor stalls
  - Possibility to configure burst type of all AHB memory transactions: INCRw or WRAPw (size w aligned on cache line size)

- **Performance counters**
  The DCACHE implements four performance counters:
  - Two hit-monitor counters (32-bit): number of read hits, number of write hits
  - Two miss-monitor counters (16-bit): number of read misses, number of write misses
- Error management
  - Possibility to detect error for master port request initiated by DCACHE itself (a cache line written back into main memory, because of an eviction or a clean operation), to flag this error, and optionally to raise an interrupt

- TrustZone security support

- Maintenance operations
  - Cache invalidate: full cache invalidation, fast command, non interruptible
  - Cache invalidate range: invalidates cache lines (reset valid bit = 0) whose address belongs to defined range, background task, interruptible
  - Cache clean range: cleans cache lines (if dirty bit = 1, write back line, then clear dirty bit) whose address belongs to defined range, background task, interruptible
  - Cache clean and invalidate range: cleans and invalidates cache lines (if dirty bit = 1, write back line, then clear valid bit) whose address belongs to defined range, background task, interruptible

9.3 DCACHE implementation

The DCACHE1 is placed on Cortex-M33 S-AHB bus and caches only the external RAM memory region (OCTOSPI and FMC), in address range [0x6000 0000:0x9FFF FFFF] of the memory map.

Indeed, by placing a bus matrix demultiplexing node in front of the DCACHE1, S-AHB bus memory requests addressing SRAM region or peripherals region (respectively in ranges [0x2000 0000:0x3FFF FFFF] and [0x4000 0000:0x5FFF FFFF]) are routed directly to the main AHB bus matrix, and the DCACHE1 is bypassed.

<table>
<thead>
<tr>
<th>Features</th>
<th>DCACHE1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of ways</td>
<td>2</td>
</tr>
<tr>
<td>Cache size</td>
<td>4 Kbytes</td>
</tr>
<tr>
<td>Cache line width</td>
<td>16 bytes</td>
</tr>
<tr>
<td>Data size of AHB Master interface</td>
<td>32 bits</td>
</tr>
</tbody>
</table>

9.4 DCACHE functional description

The purpose of the data cache is to cache external memory data loads and stores, coming from the processor. These accesses include the instruction fetches that may occur at an external memory address. The DCACHE manages both read and write transactions.
9.4.1 DCACHE block diagram

Figure 34. DCACHE block diagram

9.4.2 DCACHE reset and clocks

The DCACHE is clocked on the Cortex-M33 S-AHB bus clock.

When the DCACHE reset signal is released, a cache invalidate procedure is automatically launched, making the DCACHE busy (DCACHE_SR = 0x0000 0001).

When this procedure is finished:
- The DCACHE is invalidated: "cold cache", with all cache line valid, dirty and privilege bits = 0 (DCACHE must be filled up)
- DCACHE_SR = 0x0000 0002 (reflecting the cache is no more busy)
- The DCACHE is disabled: the EN bit in DCACHE_CR holds its reset state (= 0).

Note: When disabled, the DCACHE is bypassed: slave input requests are just forwarded to the master port.
9.4.3 DCACHE TAG memory

The DCACHE TAG memory contains:
- address tags, that indicate which data are contained in the cache data memories
- validity bits
- dirty bits
- privilege bits

There is one valid bit, one dirty bit, and one privilege bit per cache line (per way).

The valid bit enables/disables access to the data cache line: if the line is not valid, the data access (read or write) is performed in the main memory.

The valid bit is set when the cache line is written (refilled by either a read miss or a write-back miss).

Valid bits are reset in any of the below cases:
- after the DCACHE reset is released
- when the cache is disabled, by setting EN = 0 in DCACHE_CR (by software)
- when executing one of the DCACHE invalidate commands, setting by software CACHEINV = 0, or CACHECMD = 0b010 or 0b011 in DCACHE_CR (see Section 9.4.8).

The dirty bit indicates that the cache line has up-to-date values with respect to the main memory content (the cache has last right value, the main memory is not up to date).

The dirty bit is set when the cache line is written by a slave port write transaction (only in case of an access with write-back attribute).

Dirty bits are reset in any of the below cases:
- after the DCACHE reset is released
- when a line refill is performed on a read miss (on a write-back miss, the refilled cache line is modified by the written data, and dirty bit = 1)
- when the cache invalidation is performed
- when executing one of the DCACHE clean operations (cache line written back to the main memory), setting by software CACHECMD = 0b001 or 0b011 in DCACHE_CR (see Section 9.4.8).

The privilege bit indicates if the data is managed by a privileged entity. It is assigned according to the value of AHB privileged attribute at input slave port, for the first access to this line (it is written only during the line refill, on read miss or write-back miss).

The privilege bit holds same polarity as the privileged attribute: 1 for privileged access, 0 for unprivileged access.

Privilege bits are reset when the cache is invalidated, and after the DCACHE reset is released.
When a cacheable transaction is received at input slave port, its AHB address (HADDR_in) is split into the following fields (see the table below for B and W values):

- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to TAG memory address to check if the requested data is already available (meaning valid) inside the DCACHE.

The table below gives DCACHE main parameters for TAG memory dimensioning.

*Figure 35.* shows the functional view of TAG and data memories, for an n-way set associative DCACHE.

### Table 85. TAG memory dimensioning parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache size</td>
<td>S Kbytes = s bytes (s = 1024 x S)</td>
<td>4 Kbytes = 4096 bytes</td>
</tr>
<tr>
<td>Cache number of ways</td>
<td>n</td>
<td>2</td>
</tr>
<tr>
<td>Cache line size</td>
<td>L-byte = l-bit (l = 8 x L)</td>
<td>16-byte = 128-bit</td>
</tr>
<tr>
<td>Number of cache lines (per way)</td>
<td>LpW = s / (n x L) lines/way</td>
<td>128 lines/way</td>
</tr>
<tr>
<td>Address byte offset size</td>
<td>B = log2(L) bit</td>
<td>4-bit</td>
</tr>
<tr>
<td>Address way index size</td>
<td>W = log2(LpW) bit</td>
<td>7-bit</td>
</tr>
<tr>
<td>TAG address size</td>
<td>T = (32 - W - B) bit</td>
<td>21-bit</td>
</tr>
</tbody>
</table>
9.4.4 DCACHE enable

In order to activate the DCACHE functioning, the EN bit must be set in DCACHE_CR control register.

When DCACHE is disabled, it is bypassed and all transactions are copied from slave port to master port in the same clock cycle, and no comparison is performed with TAG address.

DCACHE is by default disabled at boot.

9.4.5 Cacheable and non-cacheable traffic

DCACHE is developed for Cortex-M33 core and caches the memory regions addressed by the AHB bus connected to it.

In addition, the AHB bus traffic to the memory regions can be cacheable or non-cacheable. An incoming memory request to DCACHE is defined as cacheable according to its AHB transaction memory lookup attribute.

In case of write transaction, the DCACHE write policy is defined as write-through or write-back according to its AHB transaction memory bufferable attribute (see the table below).

These AHB attributes depend on the memory protection unit (MPU) programming for the addressed region.
In case of non cacheable access, the DCACHE is bypassed, meaning that the AHB transaction is propagated unchanged to the master output port.

The bypass does not increase the latency of the access to the targeted memory.

In case of cacheable access, the DCACHE behaves as explained in Section 9.4.6.

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

### 9.4.6 Cacheable accesses

When the DCACHE receives a cacheable transaction from Cortex-M33, on its slave port, the DCACHE checks if the address requested is present in its TAG memory and if the corresponding cache line is valid.

For read transaction, there are three alternatives:

- The address is present inside the TAG memory, cache line is valid: cache read hit, the data is read from cache and provided to the processor in the same cycle.
- The address is not present in the TAG memory: cache read miss, the data is read from the main memory and provided to the processor, and a cache line refill is performed.
  
  The critical-word-first refill policy insures minimum wait cycles for the processor, since read data can be provided while cache is still performing cache line refill (associated latency is the latency of fetching one word from main memory).

  The kind of burst generated on the DCACHE master bus depends on HBURST bit in DCACHE_CR: either INCRw or WRAPw (w being the cache line width, in words).

  The AHB transaction attributes are also propagated from the slave input (missing) request to the master output refill request.
- The address is not present in the TAG memory but belongs to the refill burst from main memory that is currently ongoing: cache read hit as well (hit-under-miss feature).

Whatever the line refill is due to a read or write (missing) transaction, the DCACHE can provide the requested read data as soon as the data is available at its master interface, thus avoiding a miss (with data fetch from main memory).

<table>
<thead>
<tr>
<th>AHB lookup attribute</th>
<th>AHB bufferable attribute</th>
<th>Cacheability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>Read and write: non cacheable</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Read: cacheable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Write: (cacheable) write-through</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Read: cacheable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Write: (cacheable) write-back</td>
</tr>
</tbody>
</table>
For **write-back** transaction (write transaction, with write-back bufferable attribute), there are three alternatives as well:

- The address is present inside the TAG memory, cache line is valid: **cache write-back hit**, the data is written in cache.
- The address is not present in the TAG memory (or cache line is not valid): **cache write-back miss**.

First, a line allocation is performed by reading the entire cache line data from main memory. The kind of burst generated on the DCACHE master bus for this line refill depends on HBURST bit in DCACHE_CR: either INCRw or WRAPw (w being the cache line width, in words), and the AHB transaction attributes are propagated from the slave port initial request.

Once the refilled line is written in the DCACHE, the initial data provided on slave port is written in this DCACHE line (it overwrites the data part of the cache line that was refilled just before).

- The address is not present in the TAG memory but belongs to the refill burst from main memory that is currently ongoing: **cache write-back hit** as well (hit-under-miss feature).

Whatever the line refill is due to a read or write (missing) transaction, the DCACHE can write incoming data directly inside the refilled line, thus avoiding a miss (with refill from main memory).

For **write-through** transaction (write transaction, with write-through bufferable attribute), only two alternatives exist:

- The address is present inside the TAG memory, cache line is valid: **cache write-through hit**, the data is written both in cache and in main memory (through master port).
- The address is not present in the TAG memory (or cache line is not valid): **cache write-through miss**, the data incoming at slave port is written only in main memory (unlike the write-back miss, there is no line allocation and data written in cache).

In case of cache refill (due to cache miss), the DCACHE selects which cache line is written with the refill data: as a 2-way set associative cache, one line among two can be used (line pointed by the address index, in each of the two ways). The way selection is based on a pLRU-t replacement algorithm, that points, for each index, on the way candidate for the next refill.

If the cache line where the refill data must be written is already valid, the targeted cache line must be evicted first:

- If the dirty tag of this line equals 0 (clean data), the line is simply invalidated.
- If it equals 1 (dirty data), the line must be written back in the main memory.

The DCACHE generates a burst write transaction on its master port, with burst type set to INCRw (w being the cache line width, in words), and with AHB memory transaction attribute signals set as below:
- data (not instruction)
- privileged = TAG privilege bit
- write-back (even if it does not care)
- normal memory
- cacheable
- allocate (even if it does not care)
– non shareable
These AHB attributes cannot be propagated from the slave port (as it is the case for all other transactions emitted by the DCACHE) because the evicting transaction has no relation with the initial missing transaction. Setting of the AHB attributes is fixed, except for the privileged bit that is copied from the TAG privilege bit of the evicted line.

9.4.7 DCACHE security
The DCACHE implements an Armv8-M TrustZone.
DCACHE configuration registers are protected at system level.

9.4.8 DCACHE maintenance
The DCACHE features several maintenance operations that the software can programmed in DCACHE_CR control register:

- **Full invalidate**: invalidates the whole cache, non interruptible task.
The software can invalidate the whole DCACHE content by programming CACHEINV in DCACHE_CR.
When CACHEINV = 1, the DCACHE control logic sets BUSYF flag in DCACHE_SR status register, and performs the operation of cache invalidation, resetting each TAG valid bit to 0 (one valid bit per cache line). Each dirty and privilege bits are also reset during cache invalidation to prevent unknown values at next cache line validation. CACHEINV is automatically cleared.
Once the full invalidate operation is finished, the DCACHE automatically clears BUSYF flag, and sets BSYENDF in DCACHE_SR.
If enabled on this flag condition (BSYENDIE = 1 in DCACHE_IER), the DCACHE interrupt is raised. Then, the (empty) cache is available again.
This full invalidate operation is not interruptible, meaning that the cache does not treat any cacheable request while BUSYF = 1. However, non-cacheable traffic is treated (since the request address is not compared to TAG ones), the DCACHE being bypassed in the same clock cycle (same behavior as when the DCACHE is disabled).

- **Invalidate range**: invalidates a certain range of addresses in the cache, background task (interruptible).
The software can invalidate a given data region in the DCACHE by programming STARTCMD = 1 and CACHECMD = 0b010 in DCACHE_CR, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADDRR (range end address).
The DCACHE control logic then parses the whole TAG memory. If the read line address (TAG address + line index) falls in the programmed address range (DCACHE_CMDRSADDRR ≤ Line Addr ≤ DCACHE_CMDREADDRR), the
corresponding cache line is invalidated (line TAG bits cleared, valid bit = dirty bit = privilege bit = 0).

When STARTCMD is set, the DCACHE control logic sets BUSYCMDF in DCACHE_SR and launches the invalidate range operation. STARTCMD is also automatically cleared.

Once the operation is finished (all TAG memory parsed), the DCACHE automatically clears BUSYCMDF and sets CMDENDF in DCACHE_SR.

If enabled on this flag condition (CMDENDIE = 1 in DCACHE_IER), the DCACHE interrupt is raised.

During this invalidate range operation, the DCACHE is interruptible, meaning it can accept new incoming requests that take higher priority than the invalidation process. The TAG memory is accessed for invalidate range operation only if not already accessed by an external cache request. This implies that invalidate range execution is usually not performed in one go, but can be interrupted.

- **Clean range**: cleans a certain range of addresses in the cache, background task (interruptible).

Cleaning a cache line means making sure that the main memory content is up-to-date with the data which may have been modified in the cache. The clean operation consists in performing the write-back in the main memory of the cache lines that are tagged as “dirty” (the ones with TAG dirty bit set).

The software can clean a given data region in DCACHE by programming STARTCMD = 1, and CACHECMD = 0b001 in DCACHE_CR, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADDRR (range end address).

The DCACHE control logic then parses the whole TAG memory. If the read line address (TAG address + line index) falls in the programmed address range (DCACHE_CMDRSADDRR ≤ Line Addr ≤ DCACHE_CMDREADDRR), and the corresponding line is dirty, this line is cleaned, meaning the whole cache line is written-back in the memory through the DCACHE master port, and its TAG dirty bit is cleared.

When STARTCMD is set, the DCACHE control logic sets BUSYCMDF in DCACHE_SR and launches the clean range operation. STARTCMD is also automatically cleared.

Once the operation is finished (all TAG memory parsed), the DCACHE automatically clears BUSYCMDF and sets CMDENDF in DCACHE_SR.

If enabled on this flag condition (CMDENDIE = 1 in DCACHE_IER), the DCACHE interrupt is raised.

During this clean range operation, the DCACHE is interruptible, meaning it can accept new incoming requests that take higher priority than the cleaning process. The TAG memory is accessed for clean range operation only if not already accessed by an external cache request. This implies that clean range execution is usually not performed in one go, but can be interrupted.

It is under the software responsibility that no bus initiator attempts to change the content of the region being cleaned until clean range is completed. For that, the software can take advantage of BUSYCMDF flag in DCACHE_SR, and can poll this flag to prevent any spurious access to the area being cleaned.
Alternatively it can also rely on the command end flag (CMDENDF) or on the DCACHE interrupt to detect the end of the clean range execution.

- **Clean and invalidate range**: cleans and invalidates a certain range of addresses in the cache, background task (interruptible).

  This operation cleans the “dirty” cache lines that belong to the operation address range (the same as clean range operation), and also invalidates all the (valid) cache lines that belong to this address range (whether they are dirty or not).

  The software can launch this clean and invalidate range operation, by programming STARTCMD = 1, and CACHECMD = 0b011 in DCACHE_CR, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADADDR (range end address).

  This sets and clears the same flags, and potentially the same interrupt as invalidate range or clean range operations.

### 9.4.9 DCACHE performance monitoring

The DCACHE provides the following monitors for performance analysis:

- The two 32-bit read-hit and write-hit monitors count the AHB transactions at the DCACHE input (slave port) that do not generate a transaction on the DCACHE output (master port).

  These monitors also take into account all accesses whose address is present in the TAG memory, or in the refill buffer (due to a previous miss, and whose data is coming, or is soon to come, from cache master port) (see Section 9.4.6).

- The two 16-bit read-miss and write-miss monitors count the AHB transactions at the DCACHE input (slave port) that generate a transaction on the DCACHE output (master port).

  These monitors also take into account all accesses whose address is not present neither in the TAG memory, nor in the refill buffer.

Upon reaching their maximum values, the monitors do not wrap over.

The software can perform the following tasks:

- Enable/stop the read (write) hit monitor, through R(W)HITMEN in DCACHE_CR.
- Reset the read (write) hit monitor, by setting R(W)HITMRST in DCACHE_CR.
- Enable/stop the read (write) miss monitor, through R(W)MISSMEN in DCACHE_CR.
- Reset the read (write) miss monitor, by setting R(W)MISSMRST in DCACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

### 9.4.10 DCACHE boot

The DCACHE is disabled (EN = 0 in DCACHE_CR) at boot.

Once the boot is finished, the DCACHE can be enabled (software setting EN = 1 in DCACHE_CR).

### 9.5 DCACHE low-power modes

At product level, using the DCACHE reduces the power consumption by loading/storing data from/to the internal DCACHE most of the time, rather than from the bigger and then
more power consuming main memories. This reduction is even much higher, since the cached main memories are external.

9.6 DCACHE error management and interrupts

A transaction initiated on the DCACHE master port may return an error (a write attempt into a read-only memory, for instance). If the master port request was propagated from a slave port request, the error is propagated back to the slave port. If ever the master port request is initiated by the DCACHE itself (a cache line is written back into the main memory because of an eviction or a clean operation), the DCACHE receives this functional error and flags it internally by setting the ERRF flag in DCACHE_SR.

In such a case, an interrupt is generated if the corresponding interrupt enable bit is set (ERRIE = 1 in DCACHE_IER).

Another case of potential interrupt generation is at the end of a full invalidate operation: when the cache busy state is finished, the DCACHE sets BSYENDF flag in DCACHE_SR.

An interrupt is then generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in DCACHE_IER).

Last case is at the end of invalidate and/or clean range operations: when the command busy state is finished, the DCACHE sets CMDENDF flag in DCACHE_SR.

An interrupt is also generated if the corresponding interrupt enable bit is set (CMDENDIE = 1 in DCACHE_IER).

All DCACHE interrupt sources raise the same and unique interrupt signal, dcache_it, and then use the same interrupt vector.

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCACHE</td>
<td>Functional error</td>
<td>ERRF in DCACHE_SR</td>
<td>ERRIE in DCACHE_IER</td>
<td>Set CERRF to 1 in DCACHE_FCR</td>
</tr>
<tr>
<td></td>
<td>End of busy state (full invalidate finished)</td>
<td>BSYENDF in DCACHE_SR</td>
<td>BSYENDIE in DCACHE_IER</td>
<td>Set CBSYENDF to 1 in DCACHE_FCR</td>
</tr>
<tr>
<td></td>
<td>End of cache operations (address range based)</td>
<td>CMDENDF in DCACHE_SR</td>
<td>CMDENDIE in DCACHE_IER</td>
<td>Set CCMDENDF to 1 in DCACHE_FCR</td>
</tr>
</tbody>
</table>

The DCACHE also propagates all AHB bus errors (such as security issues, address decoding issues) from master port back to the S-AHB slave port.
### 9.7 DCACHE registers

#### 9.7.1 DCACHE control register (DCACHE_CR)

Address offset: 0x000  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>HBURST: output burst type for cache master port read accesses</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Can be set by software, only when EN = 0.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Master port write accesses are always done in INCR burst type.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: WRAP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: INCR</td>
<td></td>
</tr>
<tr>
<td>30:24</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>WMISSMRST: write-miss monitor reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: no effect</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: reset cache write-miss monitor</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>WHITMRST: write-hit monitor reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: no effect</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: reset cache write-hit monitor</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>WMISSMEN: write-miss monitor enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: cache write-miss monitor switched off. Stopping the monitor does not reset it.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: cache write-miss monitor enabled</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>WHITMEN: write-hit monitor enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: cache write-hit monitor switched off. Stopping the monitor does not reset it.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: cache write-hit monitor enabled</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>RMISSMRST: read-miss monitor reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: no effect</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: reset cache read-miss monitor</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>RHITMRST: read-hit monitor reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: no effect</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: reset cache read-hit monitor</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>RMISSMEN: read-miss monitor enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: cache read-miss monitor switched off. Stopping the monitor does not reset it.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: cache read-miss monitor enabled</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>RHITMEN: read-hit monitor enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: cache read-hit monitor switched off. Stopping the monitor does not reset it.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: cache read-hit monitor enabled</td>
<td></td>
</tr>
</tbody>
</table>
9.7.2 DCACHE status register (DCACHE_SR)

Address offset: 0x004

Reset value: 0x0000 0001

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

Bit 4 CMDENDF: command end flag
Clear by writing DCACHE_FCR.CCMDENDF = 1.
0: cache busy or in idle
1: CACHECMD command finished
9.7.3 DCACHE interrupt enable register (DCACHE_IER)

Address offset: 0x008
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>BUSYCMDF: command busy flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>cache not busy on a CACHECMD command</td>
</tr>
<tr>
<td>1</td>
<td>cache busy on a CACHECMD command (clean and/or invalidate an address range)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>ERRF: cache error flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no error</td>
</tr>
<tr>
<td>1</td>
<td>an error occurred during the operation (eviction or clean operation write-back error)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>BSYENDF: full invalidate busy end flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>cache busy or in idle</td>
</tr>
<tr>
<td>1</td>
<td>full invalidate CACHEINV operation finished</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>BUSYF: full invalidate busy flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>cache not busy on a CACHEINV operation</td>
</tr>
<tr>
<td>1</td>
<td>cache executing a full invalidate CACHEINV operation</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bit 4</th>
<th>CMDENDIE: interrupt enable on command end</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled on command end</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled on command end</td>
</tr>
</tbody>
</table>

| Bit 3 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>ERRIE: interrupt enable on cache error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled on error</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled on error</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>BSYENIE: interrupt enable on busy end</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Interrupt disabled on busy end</td>
</tr>
<tr>
<td>1</td>
<td>Interrupt enabled on busy end</td>
</tr>
</tbody>
</table>

| Bit 0 | Reserved, must be kept at reset value. |
### 9.7.4 DCACHE flag clear register (DCACHE_FCR)

Address offset: 0x00C  
Reset value: 0x0000 0000

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

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

Bit 4 **CCMDENDF**: clear command end flag  
Set by software.  
0: no effect  
1: clears CMDENDF flag in DCACHE_SR  

Bit 3 Reserved, must be kept at reset value.  

Bit 2 **CERRF**: clear cache error flag  
Set by software.  
0: no effect  
1: clears ERRF flag in DCACHE_SR  

Bit 1 **CBSYENDF**: clear full invalidate busy end flag  
Set by software.  
0: no effect  
1: clears BSYENDF flag in DCACHE_SR  

Bit 0 Reserved, must be kept at reset value.

### 9.7.5 DCACHE read-hit monitor register (DCACHE_RHMONR)

Address offset: 0x010  
Reset value: 0x0000 0000

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

Bits 31:0 **RHITMON[31:0]**: cache read-hit monitor counter
9.7.6 DCACHE read-miss monitor register (DCACHE_RMMONR)
Address offset: 0x014
Reset value: 0x0000 0000

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 RMISSMON[15:0]: cache read-miss monitor counter

9.7.7 DCACHE write-hit monitor register (DCACHE_WHMONR)
Address offset: 0x020
Reset value: 0x0000 0000

Bits 31:0 WHITMON[31:0]: cache write-hit monitor counter

9.7.8 DCACHE write-miss monitor register (DCACHE_WMMONR)
Address offset: 0x024
Reset value: 0x0000 0000

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 WMISSMON[15:0]: cache write-miss monitor counter
9.7.9 DCACHE command range start address register (DCACHE_CMDRSTARTADDR)

Address offset: 0x028
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:4</th>
<th>CMDSTARTADDR[31:4]: start address of range to which the cache maintenance command specified in DCACHE_CR.CACHECMD field applies</th>
</tr>
</thead>
</table>
| Bits 3:0 | Reserved, must be kept at reset value.

9.7.10 DCACHE command range end address register (DCACHE_CMDRENDADDR)

Address offset: 0x02C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:4</th>
<th>CMDENDADDR[31:4]: end address of range to which the cache maintenance command specified in DCACHE_CR.CACHECMD field applies</th>
</tr>
</thead>
</table>
| Bits 3:0 | Reserved, must be kept at reset value.

9.7.11 DCACHE register map

Table 88. DCACHE register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>DCACHE_CR</td>
<td>HBLRST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WMISSMRST</td>
<td>WHITMRST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WMISSMEN</td>
<td>WHITMEN</td>
<td>RMISSMRST</td>
<td>RHTMRST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RMISSMEN</td>
<td>RHITMEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 88. DCACHE register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x004</td>
<td>DCACHE_SR</td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>DCACHE_IER</td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C</td>
<td>DCACHE_FCR</td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>DCACHE_RHMONR</td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>DCACHE_RMMONR</td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x020</td>
<td>DCACHE_WHMONR</td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x024</td>
<td>DCACHE_WMMONR</td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x028</td>
<td>DCACHE_CMDRSADDR</td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x02C</td>
<td>DCACHE_CMDREADDR</td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for the register boundary addresses.
10  **Power control (PWR)**

10.1  **Introduction**

The power controller manages the device power supplies and power modes transitions.

10.2  **PWR main features**

The power controller (PWR) main features are:

- Power supplies and supply domains
  - Core domain ($V_{\text{CORE}}$)
  - $V_{\text{DD}}$ domain
  - Backup domain ($V_{\text{SW}}$)
  - Analog domain ($V_{\text{DDA}}$)
  - Supply for the SMPS power stage (available on SMPS packages)
  - $V_{\text{DDIO2}}$ supply for ten I/Os (PD6, PD7, PG9:14, PB8, PB9)
  - $V_{\text{DDUSB}}$ for USB transceiver
- System supply voltage regulation
  - SMPS step-down converter
  - Linear voltage regulator (LDO)
- Power supply supervision
  - POR/PDR monitor
  - BOR monitor
  - PVD monitor
  - AVD monitor
  - Out of functional range temperature monitor
  - Out of functional range Backup domain voltage monitor
- Power management
  - Operating modes
  - Voltage scaling control
  - Low-power modes
- $V_{\text{BAT}}$ battery charging
- TrustZone security and privileged protection
10.3 PWR pins and internal signals

Table 89. PWR input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>Supply</td>
<td>Main supply</td>
</tr>
<tr>
<td>GND</td>
<td>Supply</td>
<td>Main ground</td>
</tr>
<tr>
<td>VDDA</td>
<td>Supply</td>
<td>Analog peripherals supply</td>
</tr>
<tr>
<td>VSSA</td>
<td>Supply</td>
<td>Analog peripherals ground</td>
</tr>
<tr>
<td>VDDIO2</td>
<td>Supply</td>
<td>Independent I/O supply</td>
</tr>
<tr>
<td>VDDUSB</td>
<td>Supply</td>
<td>USB supply</td>
</tr>
<tr>
<td>VCAP</td>
<td>Supply</td>
<td>Logic supply ($V_{CORE}$)</td>
</tr>
<tr>
<td>VBAT</td>
<td>Supply</td>
<td>Backup domain supply</td>
</tr>
<tr>
<td>VDDSMPS</td>
<td>Supply</td>
<td>SMPS supply</td>
</tr>
<tr>
<td>VSSSMPS</td>
<td>Supply</td>
<td>SMPS ground</td>
</tr>
<tr>
<td>VLXSMPS</td>
<td>Supply</td>
<td>SMPS output</td>
</tr>
<tr>
<td>VREF+</td>
<td>Supply</td>
<td>ADC/DAC high reference voltage</td>
</tr>
<tr>
<td>VREF-</td>
<td>Supply</td>
<td>ADC/DAC low reference voltage</td>
</tr>
<tr>
<td>WKUPx (x = 1 to 8)</td>
<td>Input</td>
<td>Wake-up pins</td>
</tr>
<tr>
<td>CSLEEP</td>
<td>Output</td>
<td>MCU in Sleep mode</td>
</tr>
<tr>
<td>CDSTOP</td>
<td>Output</td>
<td>CPU in Stop modes</td>
</tr>
</tbody>
</table>

Table 90. PWR internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKUPx (x = 1 to 8)</td>
<td>Input</td>
<td>Wake-up event source</td>
</tr>
</tbody>
</table>
10.4 PWR power supplies and supply domains

Figure 36. Power supply with SMPS
10.4.1 External power supplies

The devices require a 1.71 to 3.6 V $V_{DD}$ operating voltage supply. Several independent supplies can be provided for specific peripherals, but must not be provided without a valid operating supply on the VDD pin:

- $V_{DD} = 1.71$ to 3.6 V
  
  $V_{DD}$ is the external power supply for the I/Os, the internal regulator, and the system analog (such as reset, power management, and internal clocks). It is provided externally through the VDD pins.

- $V_{DDA} = 1.62$ V (ADCs, DACs) / 2.1 (VREFBUF) to 3.6 V
  
  $V_{DDA}$ is the external analog power supply for A/D converters, D/A converters, and voltage reference buffer. The $V_{DDA}$ voltage level is independent from the $V_{DD}$ voltage, and must preferably be connected to $V_{DD}$ when these peripherals are not used.

- $V_{DDSMPS} = 1.71$ to 3.6 V
  
  $V_{DDSMPS}$ is the external power supply for the SMPS step-down converter. It is provided externally through the VDDSMPS supply pin, and must be connected to the same supply as the VDD pin.

- $V_{LXSMPS}$ is the switched SMPS step-down converter output.
Note: The SMPS power supply pins are available only on specific packages, with SMPS step-down converter option.

- **VDDUSB** = 3.0 to 3.6 V
  - VDDUSB is the external independent power supply for USB transceivers. The VDDUSB voltage level is independent from the VDD voltage and must preferably be connected to VDD when the USB is not used.

- **VDDIO2** = 1.08 to 3.6 V
  - VDDIO2 is the external power supply for 10 I/Os (PD6, PD7, PG9:14, PB8, PB9). The VDDIO2 voltage level is independent from the VDD voltage and must preferably be connected to VDD when those pins are not used.

- **Vcap** = 1.0V to 1.35V: digital core domain supply
  - When the voltage regulator is enabled, Vcore is delivered by the internal voltage regulator.
  - When the voltage regulator is disabled, Vcore is delivered by an external power supply through VCAP pin, or by the SMPS.

- **VBAT** = 1.62 to 3.6 V
  - VBAT is the power supply when VDD is not present (through power switch) for RTC, external clock 32 kHz oscillator, backup registers, and, optionally, backup SRAM.

- **VREF-, VREF+**
  - VREF+ is the input reference voltage for ADCs and DACs. It is also the output of the internal voltage reference buffer when enabled.
  - VREF+ can be grounded when ADC and DAC are not active.
  - The internal voltage reference buffer supports four output voltages, configured with VRS bit in the VREFBUF_CSR register:
    - VREF+ ~ 1.8 V (requires VDDA ≥ 2.1 V)
    - VREF+ ~ 2.048 V (requires VDDA ≥ 2.4 V)
    - VREF+ ~ 2.5 V (requires VDDA ≥ 2.8 V)
  - VREF- and VREF+ pins are not available on all packages. When not available, they are bonded to VSSA and VDDA, respectively.
  - When the VREF+ is double-bonded with VDDA in a package, the internal voltage reference buffer is not available and must be kept disabled.
  - VREF- must always be equal to VSSA.

### 10.4.2 Internal regulators

The devices embed two regulators: one LDO or one SMPS to provide the Vcore supply for digital peripherals, SRAMs (except BKPSRAM), and embedded flash memory. The SMPS generates this voltage on VCAP (two pins), with a total external capacitor of 10 µF (typical) and requires an external coil of 2.2 µH (typical).

The LDO generates this voltage on VCAP, with a total external capacitance of 4.7 µF (typical).

Both regulators can provide four different voltages (voltage scaling), and can operate in Stop modes.
Depending on the package configuration (SMPS or LDO), the regulator is selected by hardware. SMPS and LDO regulator are exclusively selected. The supply configurations shown in Figure 38 are supported for the V\textsubscript{CORE} domain supply.

**Figure 38. System supply configurations**

**Startup with V\textsubscript{CORE} provided from an external supply (Bypass)**

When supplied in Bypass mode, V\textsubscript{CORE} must first settle at a default level (≥ 1.1 V), before V\textsubscript{DD} reaches POR threshold level.

Due to the LDO default state after power-up (enabled by default), the external V\textsubscript{CORE} voltage must remain higher than 1.1 V until the LDO is disabled by software. When the LDO is disabled, the external V\textsubscript{CORE} voltage can be adjusted according to the user application needs (refer to section General operating conditions of the datasheet for details on V\textsubscript{CORE} level versus the maximum operating frequency).

When operating in Bypass mode, the application must adjust VOS, using bits VOS[1:0] in PWR\_VOSCR register. VOS[1:0] must be set according to the external provided core voltage level and related performance.
To adjust the VOS level, the software must select sequentially the intermediate levels.

- When increasing the performance:
  - First, voltage scaling must be incremented (for example when changing from VOS3 to VOS0, lower levels must be selected in the VOS[1:0] bits: VOS2, VOS1, and then VOS0).
  - The external voltage can be increased
  - The system frequency can be increased

- When decreasing the performance:
  - The system frequency must be decreased
  - The external voltage must be decreased
  - The voltage scaling can be decremented (for example when changing from VOS1 to VOS3, lower levels must be selected in the VOS[1:0] bits: VOS2, and then VOS3)

10.4.3 Power-up and power-down power sequences

During power-up and power-down phases, the following power sequence requirements must be respected:

- When \( V_{DD} \) is below 1 V, other power supplies \( (V_{DDA}, V_{DDIO2}, V_{DDUSB}) \) must remain below \( V_{DD} + 300 \) mV.
- When \( V_{DD} \) is above 1 V, all power supplies are independent.

During the power-down phase, \( V_{DD} \) can temporarily become lower than other supplies only if the energy provided to the MCU remains below 1 mJ. This allows external decoupling capacitors to be discharged with different time constants during the power-down transient phase.

10.4.4 Independent analog peripherals supply

To improve ADC and DAC conversion accuracy and to extend the supply flexibility, the analog peripherals have an independent power supply that can be separately filtered and shielded from noise on the PCB:

- The voltage supply input of the analog peripherals is available on a separate VDDA pin.
- An isolated supply ground connection is provided on the VSSA pin.

The \( V_{DDA} \) supply voltage can be different from \( V_{DD} \). The presence of \( V_{DDA} \) must be checked before enabling any of the analog peripherals supplied by \( V_{DDA} \) (A/D converter, D/A converter, voltage reference buffer).

Power supply level monitoring is available on VDDA via AVDO bit in PWR_VMSR register.

When a single supply is used, \( V_{DDA} \) can be externally connected to \( V_{DD} \) through the external filtering circuit to ensure a noise-free \( V_{DDA} \) reference voltage.

ADC and DAC reference voltage

To ensure a better accuracy on low-voltage inputs and outputs, the user can connect to \( V_{REF+} \), a separate reference voltage lower than \( V_{DDA} \). \( V_{REF+} \) is the highest voltage, represented by the full scale value, for an analog input (ADC) or output (DAC) signal.

\( V_{REF+} \) can be provided either by an external reference of by an internal buffered voltage reference (VREFBUF). The internal voltage reference can output a configurable voltage,
namely 1.8, 2.048 or 2.4 V. The internal voltage reference can also provide the voltage to external components through VREF+ pin. Refer to the device datasheet and to Section 29: Voltage reference buffer (VREFBUF) for further information.

Note: The VREF+ and VREF- pins are not available on all packages (internally connected, respectively, to VDDA and VSSA). Do not enable the internal voltage reference buffer when an external power supply is applied to the VREF+ pin.

10.4.5 Independent I/O supply rail

Some I/Os (PD6, PD7, PG9:14, PB8, PB9) are supplied from a separate supply rail. The power supply for this rail can range from 1.08 to 3.6 V, and is provided externally through the VDDIO2 pin. The VDDIO2 voltage level is completely independent from VDD or VDDA. The VDDIO2 pin is available only for some packages. Refer to the pinout diagrams or tables in the related device datasheet(s) for the I/O list(s).

Power supply level monitoring is available on VDDIO2 via VDDIO2RDY bit in PWR_VMSR register.

10.4.6 Independent USB transceivers supply

The USB transceivers are supplied from a separate VDDUSB power supply pin. VDDUSB range is from 3.0 to 3.6 V and is completely independent from VDD or VDDA.

Power supply level monitoring is available on VDDUSB via the USB33RDY bit in PWR_VMSR register.

Before setting USB33SV bit in PWR USB supply control register (PWR_USBSCR), check that VDDUSB is available by monitoring USB33RDY bit in PWR voltage monitor status register (PWR_VMSR). The VDD33USB supply level detector must be enabled through USB33DEN bit in PWR USB supply control register (PWR_USBSCR).

Setting USB33SV bit is mandatory to use the USBFS peripheral. It is used to validate the VDDUSB supply for electrical and logical isolation purposes.

10.4.7 Backup domain

To retain the content of the backup registers and to supply the RTC function when VDD is turned off, the VBAT pin can be connected to an optional backup voltage supplied by a battery or by another source.

The VBAT pin powers the RTC unit, the LSE oscillator, the PI8 and PC13 to PC15 I/Os, allowing the RTC to operate even when the main power supply is turned off. The backup SRAM is optionally powered by VBAT pin when the BREN bit is set in the PWR Backup domain control register (PWR_BDCR). The switch to the VBAT supply is controlled by the power-down reset embedded in the Reset block.

Warning: During tRSTTEMPO (temporization at VDD startup) or after a PDR has been detected, the power switch between VBAT and VDD remains connected to VBAT. During the startup phase, if VDD is established in less than tRSTTEMPO (refer to the datasheet for the value of tRSTTEMPO) and VDD > VBAT + 0.6 V, a current may be injected into VBAT through an internal diode connected between VDD and the...
power switch ($V_{BAT}$).
If the power supply/battery connected to the VBAT pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the VBAT pin.

---

If no external battery is used in the application, it is recommended to connect $V_{BAT}$ externally to $V_{DD}$ with a 100 nF external ceramic decoupling capacitor.

When the Backup domain is supplied by $V_{BAT}$ (analog switch connected to $V_{BAT}$), the following pins are available:

- PC13, PI8, PC14 and PC15, which can be configured by RTC or LSE (refer to Section 30.3: RTC functional description)
- PC13, PI8, PA0, PA1, and PA2 when they are configured by TAMP peripheral as tamper pins.

*Note:* As the analog switch can transfer only a limited amount of current, the use of GPIOs PC13 to PC15 and PI8 in output mode is restricted: the speed must be limited to 2 MHz with a maximum load of 30 pF, and these I/Os must not be used as a current source (for example to drive a LED).

---

**Backup domain access**

After a system reset, the Backup domain (RCC Backup domain control register RCC_BDCR, RTC registers, TAMP registers, backup registers, and backup SRAM) is protected against possible unwanted write accesses. To enable access to the Backup domain, set the DBP bit in the **PWR Backup domain control register (PWR_BDCR)** to enable access to the Backup domain.

**Backup RAM**

The Backup domain includes 4 Kbytes of backup RAM accessible in 32-, 16-, or 8-bit data mode. The backup RAM is supplied from the backup regulator in the Backup domain. When the backup regulator is enabled through BREN bit in the PWR_BDCR, the backup RAM content is retained in Standby and/or VBAT mode (it can be considered as an internal EEPROM if VBAT is always present).

The backup regulator can be ON or OFF depending whether the application needs the backup RAM function in Standby or VBAT modes.

The backup RAM is read protected and mass erased when a tamper event occurs, this is to prevent confidential data (such as a cryptographic private key) from being accessed.

The backup RAM can be erased in the following ways:

- through the flash interface after a full product state regression
- after a tamper event
- after a Backup domain reset

**$V_{BAT}$ battery charging**

When $V_{DD}$ is present, it is possible to charge the external battery on VBAT through an internal resistance.
The \( V_{BAT} \) charging is done either through a 5 k\( \Omega \) or a 1.5 k\( \Omega \) resistor, depending upon the VBRS bit value in the PWR_BDCR register.

The battery charging is enabled by setting VBE bit in the PWR_BDCR register. It is automatically disabled in \( V_{BAT} \) mode.

10.5 **PWR system supply voltage regulation**

10.5.1 **SMPS and LDO embedded regulators**

The devices embed two internal regulators, exclusively enabled by hardware, depending upon package configuration. The regulator is enabled on power-on reset. To supply the \( V_{CORE} \) from external source, it is possible to disable the regulator by setting BYPASS bit in the PWR_SCCR register.

The BYPASS bit is written once after power-on reset. Written-once mechanism locks the register and any further write access is ignored. The system must be power cycled before writing a new value.

When \( V_{CORE} \) is supplied from an external source the externally applied voltage level must be reflected in the VOSx bits in the PWR_VOSCR register.

Both regulators can provide four different voltages (voltage scaling) and can operate in Stop modes.

10.5.2 **\( V_{CORE} \) supply versus reset, voltage scaling, and low-power modes**

After reset, the \( V_{CORE} \) is in VOS3.

When exiting the Stop or Standby mode, the voltage range is the VOS3.

10.5.3 **Embedded voltage regulator operating modes**

There are three different power modes: Run, Stop, and Standby modes.

**Run mode**

The voltage regulator (LDO or SMPS) provides full power to the \( V_{CORE} \) domain (core, memories, and digital peripherals). The regulator output voltage (LDO or SMPS) can be scaled by software to different voltage levels (VOS0, VOS1, VOS2, and VOS3) that are configured through the VOS bits in the PWR voltage scaling control register (PWR_VOSCR).

The VOS voltage scaling allows optimization of the power consumption when the system is clocked below the maximum frequency. By default, VOS3 is selected after system reset. VOSx bits can be changed on-the-fly to adapt to the required system performance.

**Stop mode**

The voltage regulator (LDO or SMPS) supplies the \( V_{CORE} \) domain to retain the content of registers and internal memories. The regulator mode is selected through the SVOS bits in the PWR power mode control register (PWR_PMCR).

Stop mode power consumption can be further reduced using SVO4 (lower voltage level than VOS3) and even further with SVOS5.
Standby mode

The regulator (LDO or SMPS) is OFF and the $V_{\text{CORE}}$ domains are powered down. The content of the registers and memories is lost except for the Standby circuitry and the Backup domain.

10.6 PWR power supply and temperature supervision

Power supply level monitoring is available on the following supplies:

- $V_{\text{DD}}$ via POR/PDR (see Section 10.6.1), BOR (see Section 10.6.2), and PVD monitor (see Section 10.6.3)
- $V_{\text{DDA}}$ via AVD monitor see (Section 10.6.4)
- $V_{\text{BAT}}$ via $V_{\text{BAT}}$ threshold (see Section 10.6.6)
- $V_{\text{DDIO2}}$ via VDDIO2RDY bit (see Section 10.11.10)
- Temperature monitoring (see Section 10.6.7)

10.6.1 Power-on reset (POR)/power-down reset (PDR)

The system has an integrated POR/PDR circuitry that ensures proper startup operation. The system remains in reset mode when $V_{\text{DD}}$ is below a specified $V_{\text{POR}}$ threshold, without the need for an external reset circuit. Once the supply level is above the $V_{\text{POR}}$ threshold, the system is taken out of reset (see Figure 39). For more details concerning the reset thresholds refer to the electrical characteristics section of the datasheets.

Figure 39. Power-on (POR) / power-down (PDR) reset waveform

1. For thresholds and hysteresis values refer to the datasheets.

10.6.2 Brownout reset (BOR)

During power-on, the brownout reset (BOR) keeps the system under reset until the $V_{\text{DD}}$ supply voltage reaches the specified $V_{\text{BOR}}$ threshold.
The $V_{BOR}$ threshold is configured through system option bytes:

- BOR OFF ($BORH\_EN = 0$)
- BOR level 1 ($V_{BOR1}$)
- BOR level 2 ($V_{BOR2}$)
- BOR level 3 ($V_{BOR3}$)

By default, BOR is OFF, it can be enabled by setting $BORH\_EN$ option bit.

For more details on the brownout reset thresholds, refer to the section “Electrical characteristics” of the product datasheets.

A system reset is generated when the BOR is enabled and $V_{DD}$ supply voltage drops below the selected $V_{BOR}$ threshold.

BOR can be disabled by programming the $BORH\_EN$ option bit to 0. To disable the BOR function, $V_{DD}$ must have been higher than the POR threshold to start the system option byte programming sequence. Once BOR is disabled, the power-down is monitored by the PDR.

10.6.3 Programmable voltage detector (PVD)

The PVD can be used to monitor the $V_{DD}$ power supply by comparing it to a threshold selected by the PLS[2:0] bits in the PWR voltage monitor control register ($PWR\_VMCR$).

The PVD can also be used to monitor a voltage level on the PVD_IN pin. In this case PVD_IN voltage is compared to the internal VREFINT level.

The PVD is enabled by setting the PVDE bit in PWR voltage monitor control register ($PWR\_VMCR$).

A PVDO flag is available in the PWR voltage monitor status register ($PWR\_VMSR$) to indicate if $V_{DD}$ or PVD_IN voltage is higher or lower than the PVD threshold. This event is internally connected to the EXTI and can generate an interrupt, provided it has been enabled through the EXTI registers. The rising/falling edge sensitivity of the EXTI line must be configured according to PVD output behavior. As an example, if the EXTI line is configured to rising edge sensitivity, the interrupt is generated when $V_{DD}$ or PVD_IN voltage drops below the PVD threshold. The service routine can then start an emergency shutdown.
10.6.4 Analog voltage detector (AVD)

The AVD can be used to monitor the $V_{DDA}$ supply by comparing it to a threshold selected by the ALS[1:0] bits in the \textit{PWR voltage monitor control register (PWR_VMCR)}.

The AVD is enabled by setting the AVDEN bit in \textit{PWR voltage monitor control register (PWR_VMCR)}.

An AVDO flag is available in the \textit{PWR voltage monitor status register (PWR_VMSR)} to indicate whether $V_{DDA}$ is higher or lower than the AVD threshold. This event is internally connected to the EXTI and can generate an interrupt if enabled through the EXTI registers. The AVDO interrupt can be generated when $V_{DDA}$ drops below the AVD threshold and/or when $V_{DDA}$ rises above the AVD threshold, depending on EXTI rising/falling edge configuration. As an example, the service routine can indicate when the $V_{DDA}$ supply drops below a minimum level.
10.6.5 **V\textsubscript{DDIO2} voltage monitor (IO2VM)**

The IO2VM monitors the independent supply voltage V\textsubscript{DDIO2} to ensure that the peripheral is in its functional supply range. The VDDIO2RDY flag (see *PWR voltage monitor control register (PWR\_VMCR)*) indicates whether a valid supply is present or not.

10.6.6 **Backup domain voltage monitoring**

In VBAT mode, the battery voltage supply (backup domain) can be monitored by comparing it with two threshold levels: V\textsubscript{BAT}\textsubscript{high} and V\textsubscript{BAT}\textsubscript{low}. The VBAT supply monitoring can be enabled/disabled via MONEN bit in *PWR Backup domain control register (PWR\_BDCR)*. When it is enabled, the battery voltage thresholds increase power consumption.

If the Backup domain voltage monitoring internal tamper is enabled in the TAMP peripheral (ITAMP1E = 1 in the TAMP\_CR1 register), a tamper event is generated when the battery voltage is above the functional range.

*Note:* *The Backup domain voltage is V\textsubscript{DD} when present, V\textsubscript{BAT} otherwise.*
10.6.7 Temperature monitoring

A dedicated temperature sensor cell is embedded in the power control. The junction temperature can be monitored by comparing it with two threshold levels, TEMPhigh and TEMPlow. TEMPh and TEMPL flags in the PWR Backup domain status register (PWR_BDSR), which indicates whether the device temperature is higher or lower than the threshold. The temperature monitoring can be enabled/disabled via MONEN bit in PWR Backup domain control register (PWR_BDCR).

When enabled, the temperature thresholds increase power consumption. As an example the levels may be used to trigger a routine to perform temperature control tasks.

If the temperature monitoring internal tamper is enabled in the TAMP peripheral (ITAMP2E = 1 in the TAMP_CR1 register), a tamper event is generated when the temperature is above or below the functional range.

TEMPH and TEMPL wake-up interrupts are available on the RTC tamper signals (see Section 47: Tamper and backup registers (TAMP)).
10.7 PWR management

10.7.1 Voltage scaling

The voltage regulator supporting voltage scaling with the following features:

- Run mode voltage scaling
  - VOS0: scale 0
  - VOS1: scale 1
  - VOS2: scale 2
  - VOS3: scale 3

- Stop mode voltage scaling
  - SVOS3: scale 3
  - SVOS4: scale 4
  - SVOS5: scale 5

For more details on voltage scaling values, refer to the product datasheets.

After reset, the system starts on the lowest Run mode voltage scaling (VOS3). The voltage scaling can then be changed on-the-fly by software by programming VOS bits in PWR_VOSCR register according to the required system performance. When exiting from the Stop mode or Standby mode, the Run mode voltage scaling is reset to the default VOS3 value.

Before entering Stop mode, the software should preselect the SVOS level in PWR power mode control register (PWR_PMCR). The Stop mode voltage scaling for SVOS4 and SVOS5 also sets the voltage regulator in Low-power mode, to further reduce power consumption.

1. For thresholds and hysteresis values, refer to the datasheets.
10.7.2 Power management examples

Example of $V_{\text{CORE}}$ voltage scaling behavior in Run mode.

Figure 45. Dynamic voltage scaling in Run mode

Figure 45 illustrates the following system operation sequence example:

1. After reset, the system starts from HSI with VOS3.
2. The system performance is first increased to a medium-speed clock from the PLL with voltage scaling VOS2. To do this:
   a) Program the voltage scaling to VOS2.
   b) Once the $V_{\text{CORE}}$ supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.
3. The system performance is then increased to high-speed clock from the PLL with voltage scaling VOS1. To do this:
   a) Program the voltage scaling to VOS1.
   b) Once the VCORE supply has reached the required level indicated by VOSRDY, increase the clock frequency.
4. The system performance is then reduced to a medium-speed clock with voltage scaling VOS2. To do this:
   a) First decrease the system frequency.
   b) Then decrease the voltage scaling to VOS2.
5. The next step is to reduce the system performance to the HSI clock with voltage scaling VOS3. To do this:
   a) Switch the clock to HSI.
   b) Disable the PLL.
   c) Decrease the voltage scaling to VOS3.

6. The system performance can then be increased to high-speed clock from the PLL. To do this:
   a) Program the voltage scaling to VOS1.
   b) Once the V\textsubscript{CORE} supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.

When the system performance (clock frequency) is changed, VOS must be set accordingly, otherwise the system can be unreliable.

### 10.8 Power modes

By default, the microcontroller is in Run mode after a system or a power reset. Several low-power modes are available to save power when there is no need to keep the CPU running, for example when waiting for an external event. The user can select the mode that gives the best compromise between low-power consumption, short startup time, and wake-up sources.

The device features the following low-power modes:

- **Sleep mode**
  
  CPU clock off, all peripherals including Cortex-M33 core such as NVIC and SysTick can run and wake-up the CPU when an interrupt or an event occurs. Refer to Section 10.8.4: Sleep mode.

- **Stop mode**
  
  Stop mode achieves the lowest power consumption, while retaining the content of SRAM and registers. All clocks in the core domain are stopped. The PLL, the HSE crystal oscillators, HSI (except if HSIKERON is set), HSI48 and CSI RC (except if CSIKERON is set) are disabled. The LSE or LSI is still running. The RTC can remain active (Stop mode with RTC, Stop mode without RTC).
  
  The system clock when exiting from Stop mode can be either HSI up to 64 MHz or CSI, depending on software configuration. Refer to Section 10.8.5: Stop mode.

- **Standby mode**
  
  This mode achieves the lowest power consumption with BOR. The internal regulator is switched off so that the core domain is powered off. The PLL, the HSI RC, HSI48, the CSI RC, and the HSE crystal oscillators are also switched off. The RTC can remain active (Standby mode with RTC, Standby mode without RTC).
  
  The brownout reset (BOR) always remains active in Standby mode.
  
  The state of the I/O (except I/Os used by standby mode) during Standby mode can be retained.
  
  After entering Standby mode, SRAMs and register contents are lost except for registers...
and backup SRAM in the Backup domain and Standby circuitry.
The device exits Standby mode when an external reset (NRST pin), an IWDG reset, WKUP pin event (configurable rising or falling edge), an RTC event occurs (alarm, periodic wake-up, timestamp), or a tamper detection. The tamper detection can be raised either due to external pins or due to an internal failure detection. The system clock after wake-up is HSI at 32 MHz.
Refer to Section 10.8.6: Standby mode.
Table 91 shows the power modes overview.

### Table 91. Low-power mode summary

<table>
<thead>
<tr>
<th>Mode name</th>
<th>Entry</th>
<th>Wake-up source(1)</th>
<th>Wake-up system clock</th>
<th>Effect on clocks</th>
<th>Voltage regulators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep (Sleep-now or Sleep-on-exit)</td>
<td>WFI or Return from ISR</td>
<td>Any interrupt</td>
<td>Same as before entering Sleep mode</td>
<td>– CPU clock OFF  – No effect on other clocks or analog clock sources</td>
<td>VOS3, VOS2, VOS1, or VOS0</td>
</tr>
<tr>
<td></td>
<td>WFE</td>
<td>Wake-up event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop</td>
<td>LPMS = 0 + SLEEPDEEP bit + WFI or Return from ISR or WFE</td>
<td>– Any EXTI line (configured in the EXTI registers)</td>
<td>– CSI when STOPWUCK = 1 in RCC_CFRG</td>
<td>– All clocks OFF except LSI and LSE – HSI or CSI can be enabled temporarily when requested by software</td>
<td>SVOS3, SVOS4, or SVOS5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Specific peripherals events(2)</td>
<td>– HSI with the frequency before entering Stop mode, up to 64 MHz, when STOPWUCK = 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standby</td>
<td>LPMS = 1 + SLEEPDEEP bit + WFI or Return from ISR or WFE</td>
<td>WKUP pin edge, RTC event, IWDG reset, external reset in NRST pin</td>
<td>HSI clock at 64 MHz</td>
<td>All clocks OFF except LSI and LSE</td>
<td>OFF</td>
</tr>
</tbody>
</table>

1. Refer to Table 92.
2. Peripherals able to wake-up the system from Stop mode (this is only possible when SVOS3 is selected before entering Stop mode).

### Table 92. Functionalities depending on the working mode(1)

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Run</th>
<th>Sleep</th>
<th>Stop Wake-up capability</th>
<th>Standby Wake-up capability</th>
<th>VBAT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>SVOS3</td>
<td>SVOS4</td>
<td>SVOS5</td>
</tr>
<tr>
<td>CPU</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Flash memory (2 Mbytes)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>(2)</td>
<td>-</td>
</tr>
<tr>
<td>SRAM1 (256 Kbytes)</td>
<td>γ(3)</td>
<td>γ(3)</td>
<td>O(4)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SRAM2 (64 Kbytes)</td>
<td>γ(3)</td>
<td>γ(3)</td>
<td>O(4)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SRAM3 (320 Kbytes)</td>
<td>γ(3)</td>
<td>γ(3)</td>
<td>O(4)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BKPSRAM</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FMC</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OCTOSPI1</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### Table 92. Functionalities depending on the working mode(1) (continued)

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Run</th>
<th>Sleep</th>
<th>Stop</th>
<th>Standby</th>
<th>VBAT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Available</td>
<td>Wake-up capability</td>
<td>Available</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SVOS3</td>
<td>SVOS4</td>
<td>SVOS5</td>
</tr>
<tr>
<td>Backup registers</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Brownout reset (BOR)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Programmable voltage detector (PVD)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Analog voltage detector (AVD)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>GPDMA</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High-speed internal (HSI)</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oscillator HSI48</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High-speed external (HSE)</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low-speed internal (LSI)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>Low-speed external (LSE)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Low-power RC oscillator (CSI)</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clock security system (CSS)</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clock security system on LSE</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(5)</td>
</tr>
<tr>
<td>Backup domain voltage and temperature monitoring</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(5)</td>
</tr>
<tr>
<td>RTC/TAMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of TAMP tamper pins</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>USB FS, UCPD</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>USARTx</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>Low-power UART (LPUART)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>I2Cx (x = 1,2,3,4)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>I3C1</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>HDMI_CEC</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>SPIx (x = 1..6)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>FDCANx</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDMMCx</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ethernet</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>Serial interface (SAI)</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADCx (x = 1,2)</td>
<td>O</td>
<td>O</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAC1 (2 converters)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>VREFBUF</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>Temperature sensor (DTS)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
</tbody>
</table>
In addition, the power consumption in Run mode can be reduced by slowing down the system clocks, configuring voltage scaling to lower-power ranges, and by gating the clocks to the APB and AHB peripherals when they are not used.

---

**Table 92. Functionalities depending on the working mode**

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Run</th>
<th>Sleep</th>
<th>Stop</th>
<th>Standby</th>
<th>VBAT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Available</td>
<td>Wake-up capability</td>
<td>Available</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SVOS3</td>
<td>SVOS4</td>
<td>SVOS5</td>
</tr>
<tr>
<td>Timers (TIMx)</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Low-power timer LPTIMx (x = 1..6)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>Independent watchdog (IWDG)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(5)</td>
</tr>
<tr>
<td>Window watchdog (WWDG)</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SysTick timer (SYSTICK)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Digital camera interface (DCMI)</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Parallel synchronous slave interface (PSSI)</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CORDIC coprocessor (CORDIC)</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Filter mathematical accelerator (FMAC)</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Random number generator (RNG)</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AES and secure AES (AES, SAES)</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Public key accelerator (PKA)</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>On-the-fly decryption (OTFDEC)</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HASH accelerator</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CRC calculation unit</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GPIOs</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>EXTI</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

---

1. Y = yes (enabled), O = optional (disabled by default, can be enabled by software), - = not available. HSI or CSI are available as kernel clock for peripherals only in SVOS3.
2. The memory can be configured in Low-power mode. By default, it is not in Low-power mode during Stop (SVOS3, SVOS4).
3. The SRAM clock can be gated on or off independently. By default clock is enabled in Run and Sleep mode.
4. The SRAMs can be powered on or off independently. By default, they are not in Power-off mode during Stop.
5. Wake-up with internal tamper.
6. GPIOs state can be retained during Standby mode. By default GPIOs states are not retained.
7. 8-pin capable of wake-up from Standby mode PA0, PA2, PB7, PC1, PC13, PD2, PD3 and PI8.
Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while the debug features are used. This is due to the fact that the Cortex-M33 core is no longer clocked.

However, by setting some configuration bits in the DBGMCU control registers, the software can be debugged even when using the low-power modes extensively. For more details, refer to Section 75.2.5: Debug and low-power modes.

10.8.1 Slowing down system clocks

In Run mode, the speed of the system clocks (SYSCLK, HCLK, PCLK) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down the peripherals before entering the Sleep mode.

For more details, refer to Section 11: Reset and clock control (RCC).

10.8.2 Peripheral clock gating

In Run mode, the HCLK and PCLK for individual peripherals and memories can be stopped at any time to reduce the power consumption.

To further reduce the power consumption in Sleep mode, the peripheral clocks can be disabled before executing the WFI or WFE instructions.

The peripheral clock gating is controlled by the RCC_AHBxENR and RCC_APBxENR registers.

Disabling the peripherals clocks in Sleep mode can be performed automatically by resetting the corresponding bit in the RCC_AHBxLPENR and RCC_APBxLPENR registers.

10.8.3 Low-power modes

Entering into a low-power mode

The MCU enters in low-power modes by executing the WFI (wait for interrupt), or WFE (wait for event) instructions, or when the SLEEPONEXIT bit in the Cortex-M33 system control register is set on Return fom ISR.

Entering into a low-power mode through WFI or WFE is executed only if no interrupt is pending or no event is pending.

Exiting a low-power mode

The MCU exits the Sleep or Stop mode according to how the low-power mode was entered:

- If the WFI instruction or Return from ISR was used to enter the low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the device.
- If the WFE instruction is used to enter the low-power mode, the MCU exits the low-power mode as soon as an event occurs. The wake-up event can be generated either by:
  - an NVIC IRQ interrupt:
    > When SEVONPEND = 0 in the Cortex-M33 system control register
    By enabling an interrupt in the peripheral control register and in the NVIC.
    When the MCU resumes from WFE, the peripheral interrupt pending bit and.
the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) must be cleared. Only NVIC interrupts with high enough priority can wake up and interrupt the MCU.

> When SEVONPEND = 1 in the Cortex-M33 system control register
By enabling an interrupt in the peripheral control register and optionally in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and when enabled the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) must be cleared. All NVIC interrupts wake up the MCU, even the disabled ones. Only enabled NVIC interrupts with high enough priority can wake up and interrupt the MCU.

– an event:

> Configuring an EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit, as the pending bits corresponding to the event line are not set. It may be necessary to clear the interrupt flag in the peripheral.

The MCU exits Standby mode through an external reset (NRST pin), an IWDG reset, a rising edge on one of the enabled WKUPx pins or a RTC/TAMP event (see Figure 617: RTC block diagram).

After waking up from Standby mode, the program execution restarts in the same way as after a reset (boot pin sampling, option bytes loading, reset vector is fetched).

**Caution:** When the device is in Stop mode, a peripheral interrupt powers on an internal oscillator. The corresponding NVIC interrupt channel must be enabled to allow the interrupt to exit the device from Stop mode. It is not allowed to disable a peripheral interrupt by disabling only the NVIC channel while keeping the peripheral interrupt enable, as the device could remain in Stop mode with clock ON.

### 10.8.4 Sleep mode

**I/O states in Sleep mode**

In Sleep mode, all I/O pins keep the same state as in Run mode.

**Entering the Sleep mode**

The MCU enters the Sleep mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is clear (see the table below for details on how to enter the Sleep mode).

**Exiting the Sleep mode**

The MCU exits the Sleep mode as described in *Exiting a low-power mode* (see the table below for details on how to exit the Sleep mode).
The Stop mode is based on the Cortex-M33 Deepsleep mode combined with the peripheral clock gating. The voltage regulator is configured by SVOSx bits (the selected SVOS4 and SVOS5 levels add an additional startup delay when exiting from system Stop mode). In Stop mode, all clocks in the core domain are stopped. The PLL, HSI, HSI48, CSI and HSE oscillators are disabled.

It is possible to keep the HSI or CSI clock enabled during Stop mode, to be quickly available as kernel clock for peripherals.

All SRAMs and register contents are preserved, but the SRAMs can be totally or partially switched off to further reduce consumption. The user can select which memory is discarded during Stop mode by means of xxSO bits in PWR power mode control register (PWR_PMCR).

10.8.5 Stop mode

The Stop mode is based on the Cortex-M33 Deepsleep mode combined with the peripheral clock gating. The voltage regulator is configured by SVOSx bits (the selected SVOS4 and SVOS5 levels add an additional startup delay when exiting from system Stop mode). In Stop mode, all clocks in the core domain are stopped. The PLL, HSI, HSI48, CSI and HSE oscillators are disabled.

It is possible to keep the HSI or CSI clock enabled during Stop mode, to be quickly available as kernel clock for peripherals.

All SRAMs and register contents are preserved, but the SRAMs can be totally or partially switched off to further reduced consumption. The user can select which memory is discarded during Stop mode by means of xxSO bits in PWR power mode control register (PWR_PMCR).

Table 93. Sleep mode

<table>
<thead>
<tr>
<th>Sleep mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode entry</td>
<td>WIFI (wait for interrupt) or WFE (wait for event) while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP = 0</td>
</tr>
<tr>
<td></td>
<td>– No interrupt (for WIFI) or event (for WFE) pending</td>
</tr>
<tr>
<td></td>
<td>Refer to the Cortex-M33 system control register.</td>
</tr>
<tr>
<td>Mode exit</td>
<td>On return from ISR while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP = 0 and</td>
</tr>
<tr>
<td></td>
<td>– SLEEPONEXIT = 1</td>
</tr>
<tr>
<td></td>
<td>– No interrupt pending</td>
</tr>
<tr>
<td></td>
<td>Refer to the Cortex-M33 system control register.</td>
</tr>
<tr>
<td></td>
<td>If WIFI or Return from ISR was used for entry</td>
</tr>
<tr>
<td></td>
<td>Interrupt (see Table 135: STM32H563/H573 and STM32H562 vector table)</td>
</tr>
<tr>
<td></td>
<td>If WFE was used for entry and SEVONPEND = 0:</td>
</tr>
<tr>
<td></td>
<td>Wake-up event (see Section 18.3: EXTI functional description)</td>
</tr>
<tr>
<td></td>
<td>If WFE was used for entry and SEVONPEND = 1:</td>
</tr>
<tr>
<td></td>
<td>Interrupt even when disabled in NVIC (see Table 135: STM32H563/H573 and STM32H562 vector table) or wake-up event (see Section 18.3: EXTI functional description)</td>
</tr>
<tr>
<td>Wake-up latency</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 94. Memory shut-off block selection

<table>
<thead>
<tr>
<th>Selection bit</th>
<th>Shut-off block in Stop mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM1SO</td>
<td>AHB SRAM1</td>
</tr>
<tr>
<td>SRAM2_48SO</td>
<td>AHB SRAM2 48-Kbyte</td>
</tr>
<tr>
<td>SRAM2_16SO</td>
<td>AHB SRAM2 16-Kbyte</td>
</tr>
<tr>
<td>SRAM3SO</td>
<td>AHB SRAM3</td>
</tr>
<tr>
<td>ETHERNETSO</td>
<td>ETHERNET RAM</td>
</tr>
</tbody>
</table>

The BOR is always available in Stop mode.
I/O states in Stop mode

In the Stop mode, all I/O pins keep the same state as in the Run mode.

Entering the Stop mode

The MCU enters the Stop mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see Table 95 for details on how to enter the Stop mode).

If the flash memory programming is ongoing, the Stop mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, the Stop mode entry is delayed until the APB access is finished.

In Stop mode, the following features can be selected by programming the individual control bits:

- The independent watchdog (IWDG) is started by writing to its key register or by hardware option. Once started, it can be stopped only by a reset (see Section 44: *Independent watchdog (IWDG)*).
- The real-time clock (RTC) is configured by the RTCEN bit in the *RCC backup domain control register (RCC_BDCR)*.
- The internal RC oscillator LSI clock is configured by the LSION bit in RCC_BCDR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BCDR.

The AVD and the PVD can be used in Stop mode. If they are not needed, they must be disabled by software to save their power consumptions.

The ADCx (x = 1, 2), the DAC1 (two channels), the temperature sensor, and the VREFBUF can consume power during the Stop mode, unless they are disabled before entering this mode.

Exiting the Stop mode

The MCU exits Stop mode by enabling an EXTI interrupt or event depending on how the low-power mode was entered. Some peripherals are able to wake up the system (refer to Table 138: *EXTI line connections*) from Stop mode, this is only possible when SVOS3 is selected before entering Stop mode.

*Note: When wake-up from Stop with peripherals is needed, SVOS3 must be selected.*

When exiting Stop mode by issuing an interrupt or a wake-up event, CSI is selected as system clock if bit STOPWUCK is set in RCC clock configuration register 1 (RCC_CFGR). The HSI oscillator is selected as system clock if STOPWUCK is cleared. The wake-up time is shorter when CSI is selected as wake-up system clock. The HSI selection allows a wake-up at higher frequency (up to 64 MHz).

The MCU exits Stop mode by enabling an EXTI interrupt or event depending on how the low-power mode was entered.

When exiting the Stop mode, the MCU is in Run mode, VOS3.
10.8.6 Standby mode

The lowest power mode in which the BOR is active is the Standby mode. It is based on the Cortex-M33 Deepsleep mode, with the voltage regulators disabled. The PLL, HSI, HSI48, CSI, and HSE oscillators are also switched off.

The SRAMs and register contents are lost except for registers in the Backup domain and Standby circuitry (see Figure 36: Power supply with SMPS).

The BOR is always available in Standby mode.
I/O states in Standby mode

In the Standby mode, the I/Os are by default in floating state. If the IORETEN bit in the PWR_IORETR register is set, the I/Os output state is retained. I/O retention mode is enabled for all I/Os except those supporting the standby functionality and JTAG IOs (PA13, PA14, PA15 and PB4). When entering into Standby mode, the state of the output is sampled, and pull-up or pull-down resistor are set to maintain the I/O output during Standby mode.

If the JTAGIORETEN bit in the PWR_IORETR register is set, the I/Os output state is retained. I/O retention mode is enabled for PA13, PA14, PA15 and PB4 (default JTAG pull-up/pull-down after wake-up is not enabled).

**Figure 46. I/O states in Standby mode**

<table>
<thead>
<tr>
<th>IO state retention disabled</th>
<th>GPIO mode</th>
<th>System mode</th>
<th>Standby entry</th>
<th>IORETEN (PWR)</th>
<th>Wakeup request</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
<td>Floating</td>
<td>Normal (default after reset)</td>
<td>Run</td>
<td>Standby (Vcore off)</td>
</tr>
<tr>
<td>IO state retention enabled</td>
<td>GPIO mode</td>
<td>System mode</td>
<td>Standby entry</td>
<td>IORETEN (PWR)</td>
<td>Wakeup request</td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>State retained (PU/PD)</td>
<td>Normal</td>
<td>Run</td>
<td>Standby (Vcore off)</td>
</tr>
</tbody>
</table>

After wakeup from Standby mode, as long as IORETEN (or JTAGIORETEN for JTAG IOs) is set, the retained state (pull-up/pull-down) remains applied.

The GPIO pin state before standby can be identified with GPIO_IDR register (when both GPIO port clock and input buffer are enabled).

The application can release the IO state (clear the retained Pull-up/Pull-down) by clearing the IORETEN (or JTAGIORETEN for JTAG IOs) bit, this can be done before or after reconfiguring the GPIOs and related peripherals. This lets the application configure the GPIO to a known state before releasing the retained state.

The RTC outputs on PC13 and PI8 are functional in Standby mode. PC14 and PC15 used for LSE are also functional. The 8 wake-up pins (WKUPx, x = 1 to 8) and four RTC tampers pins are available.

**Entering Standby mode**

The MCU enters the Standby mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 96* for details on how to enter Standby mode).
In Standby mode, the following features can be selected by programming individual control bits:

- The independent watchdog (IWDG) is started by writing to its Key register or by hardware option. Once started it can be stopped only by a reset (see Section 44.4: IWDG functional description).
- The real-time clock (RTC) is configured by the RTCEN bit in RCC backup domain control register (RCC_BDCR).
- The internal RC oscillator LSI clock, is configured by the LSION bit in RCC_BDCR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BDCR.
- The IOs retention is configured by the IORETEN bit in the PWR_IORETR register.

**Exiting Standby mode**

The MCU exits the Standby mode as described in Exiting a low-power mode. The SBF status flag in the PWR status register (PWR_PMSR) indicates that the MCU was in Standby mode. All registers are reset after wake-up from Standby except for PWR Backup domain control register (PWR_BDCR) and PWR I/O retention register (PWR_IORETR) (see Table 96 for more details on how to exit Standby mode).

When exiting Standby mode, I/Os output state that were retained during Standby through IORETEN bit, keep this configuration upon exiting Standby mode until the IORETEN bit in PWR_IORETR register is cleared. Once IORETEN is cleared, the I/Os are either configured to their reset values or to the pull-up/pull-down state according to the GPIOx_PUPDR registers.

For I/Os, with a pull-up or pull-down predefined after reset (some JTAG/SWD I/Os), in case those pull-up or pull-down are different from the retained values during Standby, both a pull-down and pull-up are applied until IORETEN is cleared, releasing the retained value.

Also in case the GPIOx_PUPDR values programed after exiting from Standby are different from the retained values during Standby, both a pull-down and pull-up are applied until IORETEN is cleared, releasing the retained value.

**Table 96. Standby mode**

<table>
<thead>
<tr>
<th>Standby mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode entry</td>
<td>WFI (wait for interrupt) or WFE (wait for event) while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP bit is set in Cortex-M33 system control register</td>
</tr>
<tr>
<td></td>
<td>– No interrupt (for WFI) or event (for WFE) pending</td>
</tr>
<tr>
<td></td>
<td>– LPMS = 1 in PWR_PMCR</td>
</tr>
<tr>
<td></td>
<td>– WUFx bits cleared in PWR_WUSR</td>
</tr>
<tr>
<td>On Return from ISR while:</td>
<td></td>
</tr>
<tr>
<td>– SLEEPDEEP bit is set in Cortex-M33 system control register</td>
<td></td>
</tr>
<tr>
<td>– SLEEPONEXIT = 1</td>
<td></td>
</tr>
<tr>
<td>– No interrupt pending</td>
<td></td>
</tr>
<tr>
<td>– LPMS = 1 in PWR_PMCR</td>
<td></td>
</tr>
<tr>
<td>– WUFx bits cleared in PWR_WUSR</td>
<td></td>
</tr>
<tr>
<td>– RTC/TAMP flags corresponding to the chosen wake-up source, cleared</td>
<td></td>
</tr>
</tbody>
</table>
### 10.8.7 Power modes output pins

In order to help the debug, three signals are available as device pins alternate functions:

- **CSLEEP**
  - When set, CSLEEP indicates that the system is in Sleep mode: WFI or WFE has been executed.
  - When cleared, CSLEEP indicates that the system is in Run mode.

- **CDSTOP**
  - When set, CDSTOP indicates that the system is in CStop mode, meaning that the following conditions are fulfilled:
    - WFI or WFE has been executed with CPU SLEEPDEEP = 1.
    - No AHB/APB clock is running.
  - When cleared, CDSTOP indicates that the system is not in CStop mode: AHB/APB clocks are running.

The table below explains the MCU power mode depending on these signals states.

<table>
<thead>
<tr>
<th>CSLEEP</th>
<th>CSTOP</th>
<th>MCU power modes(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Run mode</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Sleep mode or Stop mode, with AHB/APB clocks running</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Stop mode</td>
</tr>
</tbody>
</table>

1. CSLEEP and CDSTOP are generated in core domain, consequently they are not driven in Standby mode.

### 10.9 PWR security and privileged protection

#### 10.9.1 PWR security protection

When the TrustZone security is activated by the TZEN option byte in the flash memory option byte configuration register, some PWR register fields can be secured against non-secure access.

---

**Table 96. Standby mode**

<table>
<thead>
<tr>
<th>Standby mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode exit</td>
<td>WKUPx pin edge, RTC event, external Reset in NRST pin, IWDG Reset, BOR reset</td>
</tr>
<tr>
<td>Wake-up latency</td>
<td>Reset phase</td>
</tr>
</tbody>
</table>
The PWR TrustZone security allows the following features to be secured through the PWR_SECCFGR register:

- Low-power mode
- Wake-up (WKUP) pins
- Voltage detection and monitoring
- VBAT mode
- I/Os retention configuration

Other PWR configuration bits are secure when:

- The system clock selection is secure in RCC: the voltage scaling (VOS) configuration is secure.
- The UCPD1 is secure in the GTZC: the PWR_UCPDR register is secure.

Table 98 gives a summary of the PWR secured bits following the security configuration bit in PWR_SECCFGR.

A non-secure access to a secure-protected register bit is denied:

- The secured bits are not written (WI) with a non-secure write access.
- The secured bits are read as 0 (RAZ) with a non-secure read access.

A non-secure write access to PWR_SECCFGR is WI and generates an illegal access event and an interrupt if enabled in the GTZC. It can be read with a non-secure read access.

When the TrustZone security is disabled (TZEN = 0xC3), PWR_SECCFGR is RAZ/WI and all other registers are non-secure.

### Table 98. PWR security configuration summary

<table>
<thead>
<tr>
<th>Secure configuration register</th>
<th>Security configuration bit</th>
<th>Register name</th>
<th>Secured bits</th>
<th>Non-secure access on secure bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR_SECCFGR</td>
<td>Not applicable(1)</td>
<td>PWR_SECCFGR</td>
<td>All bits</td>
<td>Read OK. WI and illegal access event</td>
</tr>
<tr>
<td>PWR_SECCFGR</td>
<td>At least one bit is set</td>
<td>PWR_PRIVCFGR</td>
<td>SPRIV</td>
<td>Read OK. WI</td>
</tr>
<tr>
<td>PWR_SECCFGR</td>
<td>LPMSEC</td>
<td>PWR_PMCR</td>
<td>All bits</td>
<td>WI</td>
</tr>
<tr>
<td>PWR_SECCFGR</td>
<td>VUSBSEC</td>
<td>PWR_USBSCR</td>
<td>All bits</td>
<td>WI</td>
</tr>
<tr>
<td>PWR_SECCFGR</td>
<td>VBSEC</td>
<td>PWR_BDCR</td>
<td>All bits</td>
<td>WI</td>
</tr>
<tr>
<td>PWR_SECCFGR</td>
<td>RETSEC</td>
<td>PWR_IORETR</td>
<td>All bits</td>
<td>WI</td>
</tr>
<tr>
<td>PWR_SECCFGR</td>
<td>WUPxSEC (x = 1 to 8)</td>
<td>PWR_WUCR</td>
<td>WUPNx</td>
<td>RAZ/WI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WUPPx</td>
<td>RAZ/WI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WUPPUPD</td>
<td>RAZ/WI</td>
</tr>
<tr>
<td>GTZC_TZSC_SECCFGR</td>
<td>UCPD1SEC</td>
<td>PWR_UCPDR</td>
<td>All bits</td>
<td>RAZ/WI</td>
</tr>
</tbody>
</table>
10.9.2 PWR privileged protection

By default, after a reset, all registers can be read or written with both privileged and unprivileged accesses, except PWR_PRIVCFGR, which can be written only with privileged access. PWR_PRIVCFGR can be read by secure and non secure, privileged and unprivileged accesses.

The SPRIV bit in PWR_PRIVCFGR can be written only with secure privileged access. This bit configures the privileged access of all PWR secure functions (defined by PWR_SECCFGR, GTZC, RCC, or GPIO, as shown in Table 98).

When the SPRIV bit is set in PWR_PRIVCFGR:

- The PWR secure bits can be written only with privileged access, including PWR_SECCFGR.
- The PWR secure bits can be read only with privileged access except PWR_SECCFGR and PWR_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged PWR bit or register is discarded: the bits are read as 0 and the write to these bits is ignored (RAZ/WI).

The NSPRIV bit of PWR_PRIVCFGR can be written only with privileged access, secure or non-secure. It configures the privileged access of all PWR securable functions configured as non-secure (defined by PWR_SECCFGR, GTZC, RCC, or GPIO, see Table 98).

When the NSPRIV bit is set in PWR_PRIVCFGR:

- The PWR securable bits configured as non-secure, can be written only with privileged access.
- The PWR securable bits configured as non-secure, can be read only with privileged access except PWR_PRIVCFGR, which can be read by privileged or unprivileged accesses.
- The VOSRDY and BOOSTRDY bits in PWR_VOSR, PWR_SR, PWR_SVMSR, PWR_BDSR and PWR_WUSR, can be read with privileged or unprivileged accesses.
- An unprivileged access to a privileged PWR bit or register is discarded: the bits are read as 0, and the write to these bits is ignored (RAZ/WI).

10.10 PWR interrupts

Table 99 gives a summary of the interrupt sources and the way to control them.
### Table 99. PWR interrupt requests

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit Sleep, Stop modes</th>
<th>Exit Standby modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVD/AVD output</td>
<td>Programmable voltage detector through EXTI line 16</td>
<td>PVDO/AVDO</td>
<td>EXTI line 16 enabled</td>
<td>Write EXTI PIF16 = 1</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
10.11 **PWR registers**

The PWR registers can be accessed in word, half-word and byte format, unless otherwise specified.

10.11.1 **PWR power mode control register (PWR_PMCR)**

This register is protected against non-secure access when LPMSEC = 1 in the PWR_SECCFGR register.

This register is protected against unprivileged access when LPMSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when LPMSEC = 0 and NSPRIV = 1.

Address offset: 0x000

Reset value: 0x0000 000C

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 26 **SRAM1SO**: AHB SRAM1 shut-off in Stop mode
- 0: AHB RAM1 content is kept in Stop mode.
- 1: AHB RAM1 content is lost in Stop mode.

Bit 25 **SRAM2_48SO**: AHB SRAM2 48-Kbyte shut-off in Stop mode
- 0: AHB RAM2 48-Kbyte content is kept in Stop mode.
- 1: AHB RAM2 48-Kbyte content is lost in Stop mode.

Bit 24 **SRAM2_16SO**: AHB SRAM2 16-Kbyte shut-off in Stop mode
- 0: AHB RAM2 16-Kbyte content is kept in Stop mode.
- 1: AHB RAM2 16-Kbyte content is lost in Stop mode.

Bit 23 **SRAM3SO**: AHB SRAM3 shut-off in Stop mode
- 0: AHB RAM3 content is kept in Stop mode.
- 1: AHB RAM3 content is lost in Stop mode.

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **ETHERNETSO**: ETHERNET RAM shut-off in Stop mode
- 0: ETHERNET RAM content is kept in Stop mode.
- 1: ETHERNET RAM content is lost in Stop mode.

Bits 15:14 Reserved, must be kept at reset value.
Bit 13 AVDREADY: analog voltage ready
This bit is used only when the analog switch boost needs to be enabled (see BOOSTE bit).
It must be set by software when the expected VDDA analog supply level is available.
The correct analog supply level is indicated by the AVDO bit (PWR_VMSR register) after
setting the AVDEN bit (PWR_VMCR register) and selecting the supply level to be monitored
(ALS bits).
0: peripheral analog voltage VDDA not ready (default)
1: peripheral analog voltage VDDA ready.

Bit 12 BOOSTE: analog switch VBOOST control
This bit enables the booster to guarantee the analog switch AC performance when the VDD
supply voltage is below 2.7 V (reduction of the total harmonic distortion to have the same
switch performance over the full supply voltage range) The VDD supply voltage can be
monitored through the PVD and the PLS bits.
0: booster disabled (default)
1: booster enabled if analog voltage ready (AVDREADY = 1)

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 FLPS: flash memory low-power mode in Stop mode
This bit is used to obtain the best trade-off between low-power consumption and restart time
when exiting from Stop mode.
When it is set, the flash memory enters low-power mode when the system is in Stop mode.
0: flash memory remains in normal mode when the system enters Stop mode (quick restart
time).
1: flash memory enters low-power mode when the system enters Stop mode (low-power
consumption).

Note: When system enters Stop mode with SVOS5 enabled, flash memory is automatically
forced in low-power mode.

Bit 8 Reserved, must be kept at reset value.

Bit 7 CSSF: clear Standby and Stop flags (always read as 0)
This bit is cleared to 0 by hardware.
0: no effect
1: STOPF and SBF flags cleared

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:2 SVOS[1:0]: system Stop mode voltage scaling selection
These bits control the VCORE voltage level in system Stop mode, to obtain the best trade-off
between power consumption and performance.
00: reserved
01: SVOS5 scale 5
10: SVOS4 scale 4
11: SVOS3 scale 3 (default)

Bit 1 Reserved, must be kept at reset value.

Bit 0 LPMS: low-power mode selection
This bit defines the Deepsleep mode.
0: Keeps Stop mode when entering DeepSleep.
1: Allows Standby mode when entering DeepSleep.
### 10.11.2 PWR status register (PWR_PMSR)

Address offset: 0x004  
Reset value: 0x0000 0000

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

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

**Bits 31:7**  Reserved, must be kept at reset value.

**Bit 6 SBF**: System standby flag

This bit is set by hardware and cleared only by a POR or by setting the CSSF bit.
0: system has not been in Standby mode.  
1: system has been in Standby mode.

**Bit 5 STOPF**: Stop flag

This bit is set by hardware and cleared only by any reset or by setting the CSSF bit.
0: system has not been in Stop mode.  
1: system has been in Stop mode.

**Bits 4:0**  Reserved, must be kept at reset value.

### 10.11.3 PWR voltage scaling control register (PWR_VOSCR)

Some register fields are protected against non-secure access depending on RCC_SECCFGR register.

These fields can be protected against unprivileged access depending on PWR_PRIVCFGR register configuration.

Address offset: 0x0010

Reset value: 0x0000 0000

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

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

**Bits 31:6**  Reserved, must be kept at reset value.
Bits 5:4 **VOS[1:0]**: voltage scaling selection according to performance
These bits control the $V_{\text{CORE}}$ voltage level and allow to obtain the best trade-off between power consumption and performance:
- In bypass mode, these bits must also be set according to the external provided core voltage level and related performance.
- When increasing the performance, the voltage scaling must be changed before increasing the system frequency.
- When decreasing performance, the system frequency must first be decreased before changing the voltage scaling.
  - 00: scale 3 (default)
  - 01: scale 2
  - 10: scale 1
  - 11: scale 0

Bits 3:0 Reserved, must be kept at reset value.

### 10.11.4 PWR voltage scaling status register (PWR_VOSSR)

Address offset: 0x0014
Reset value: 0x0000 2008

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>ACTVOS[1:0]</th>
<th>ACTVORSRDY</th>
<th>VOSRDY</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **ACTVOS[1:0]**: voltage output scaling currently applied to $V_{\text{CORE}}$
This field provides the last VOS value.
- 00: VOS3 (lowest power)
- 01: VOS2
- 10: VOS1
- 11: VOS0 (highest frequency)

Bit 13 **ACTVORSRDY**: Voltage level ready for currently used VOS
- 0: $V_{\text{CORE}}$ is above or below the current voltage scaling provided by ACTVOS[1:0]
- 1: $V_{\text{CORE}}$ is equal to the current voltage scaling provided by ACTVOS[1:0]

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **VOSRDY**: Ready bit for $V_{\text{CORE}}$ voltage scaling output selection.
- 0: Not ready, voltage level below VOS selected level
- 1: Ready, voltage level at or above VOS selected level

The VOSRDY flag must be used only when switching from low to high-voltage scale (like switching from VOS3 to VOS0).

Bits 2:0 Reserved, must be kept at reset value.
10.11.5 PWR Backup domain control register (PWR_BDCR)

This register is protected against non-secure access when VBSEC = 1 in the PWR_SECCFG register.

This register is protected against unprivileged access when VBSEC = 1 and SPRIV = 1 in the PWR_PRIVCFG register, or when VBSEC = 0 and NSPRIV = 1.

This register is not reset by wake-up from Standby mode, RESET signal and VDD POR. It is only reset by VSW POR and VSWRST reset. This register must not be accessed when VSWRST bit in RCC_BDCR register resets the VSW domain.

MONEN and BREN bits must not be accessed when VSWRST bit in RCC_BDCR register resets the VSW domain. After reset, MONEN and BREN of this register are write-protected. Prior to modifying their content, the DBP bit in PWR_DBPCR register must be set to disable the write protection.

Address offset: 0x20

Power-on reset value: 0x0000 0000

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

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

Bit 9 **VBRS**: \( V_{\text{BAT}} \) charging resistor selection
0: Charge \( V_{\text{BAT}} \) through a 5 k\( \Omega \) resistor.
1: Charge \( V_{\text{BAT}} \) through a 1.5 k\( \Omega \) resistor.

Bit 8 **VBE**: \( V_{\text{BAT}} \) charging enable
0: \( V_{\text{BAT}} \) battery charging disabled.
1: \( V_{\text{BAT}} \) battery charging enabled.

Note: Reset only by POR.

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

Bit 1 **MONE N**: Backup domain voltage and temperature monitoring enable
0: Backup domain voltage and temperature monitoring disabled
1: Backup domain voltage and temperature monitoring enabled

Bit 0 **BREN**: Backup RAM retention in Standby and \( V_{\text{BAT}} \) modes
When this bit set, the backup regulator (used to maintain the backup RAM content in Standby and \( V_{\text{BAT}} \) modes) is enabled.
If BREN is cleared, the backup regulator is switched off. The backup RAM can still be used in Run and Stop modes. However its content is lost in Standby and \( V_{\text{BAT}} \) modes.
If BREN is set, the application must wait till the backup regulator ready flag (BRRDY) is set to indicate that the data written into the SRAM is maintained in Standby and \( V_{\text{BAT}} \) modes.
0: Backup RAM content lost in Standby and \( V_{\text{BAT}} \) modes.
1: Backup RAM content preserved in Standby and \( V_{\text{BAT}} \) modes.
10.11.6 **PWR Backup domain control register (PWR_DBPCR)**

This register is protected against non-secure access when VBSEC = 1 in the PWR_SECCFG register.

This register is protected against unprivileged access when VBSEC = 1 and SPRIV = 1 in the PWR_PRIVCFG register, or when VBSEC = 0 and NSPRIV = 1.

Address offset: 0x24

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:1</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>0</td>
<td><strong>DBP</strong>: Disable Backup domain write protection</td>
</tr>
<tr>
<td></td>
<td>In reset state, all registers and SRAM in Backup domain are protected against parasitic write access. This bit must be set to enable write access to these registers.</td>
</tr>
<tr>
<td></td>
<td>0: Write access to Backup domain disabled</td>
</tr>
<tr>
<td></td>
<td>1: Write access to Backup domain enabled</td>
</tr>
</tbody>
</table>

10.11.7 **PWR Backup domain status register (PWR_BDSR)**

This register is not reset by wake-up from Standby mode, RESET signal and VDD POR. It is only reset by VSW POR and VSWRST reset.

Address offset: 0x028

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23</td>
<td><strong>TEMPH</strong>: temperature level monitoring versus high threshold</td>
</tr>
<tr>
<td></td>
<td>0: temperature below high threshold level</td>
</tr>
<tr>
<td></td>
<td>1: temperature equal or above high threshold level</td>
</tr>
<tr>
<td>22</td>
<td><strong>TEMPL</strong>: temperature level monitoring versus low threshold</td>
</tr>
<tr>
<td></td>
<td>0: temperature above low threshold level</td>
</tr>
<tr>
<td></td>
<td>1: temperature equal or below low threshold level</td>
</tr>
</tbody>
</table>
10.11.8 **PWR USB Type-C power delivery register (PWR_UCPDR)**

This register is protected against non-secure access when UCPD1SEC = 1 in the TZSC_SECCFG register.

This register is protected against unprivileged access when UCPD1SEC = 1 and SPRIV = 1 in the PWR_PRIVCFG register, or when UCPD1SEC = 0 and NSPRIV = 1.

Address offset: 0x02C


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

Not affected by exit Standby mode.

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

**Bit 1 UCPD_STBY: USB Type-c and Power delivery Standby mode**

When set, this bit is used to memorize the UCPD configuration in Standby mode. This bit must be written to 1 just before entering Standby mode when using UCPD, and it must be written to 0 after exiting the standby mode and before writing any UCPD register.

**Bit 0 UCPD_DBDIS: USB Type-C and power delivery dead battery disable**

After exiting reset, the USB Type-C "dead battery" behavior is enabled, which may have a pull-down effect on CC1 and CC2 pins. It is recommended to disable it in all case, either to stop this pull-down or to hand over control to the UCPD (which should therefore be initialized before doing the disable).

0: Enable USB Type-C dead battery pull-down behavior on UCPDx_CC1 and UCPDx_CC2 pins.
1: Disable USB Type-C dead battery pull-down behavior on UCPDx_CC1 and UCPDx_CC2 pins.
10.11.9 **PWR supply configuration control register (PWR_SCCR)**

This register is protected against non-secure access when SCMSEC = 1 in the PWR_SECCFGR register.

This register is protected against unprivileged access when SCMSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when SCMSEC = 0 and NSPRIV = 1.

Address offset: 0x030

Reset value: 0x0000 0X00

The reset value of register change according to the package. The bits 9 and 8 indicate the power configuration. Their values are exclusive.

Reset by POR only, not reset by wake-up from Standby mode and RESET pad. The BYPASS bit of this register is written once after POR. Written-once mechanism locks the register and any further write access is ignored. The system must be power cycled before writing a new value.

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

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

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

- **Bit 9 SMPSEN**: SMPS enable
  - The value is set by hardware when the package uses the SMPS regulator.

- **Bit 8 LDOEN**: LDO enable
  - The value is set by hardware when the package uses the LDO regulator.

Bits 7:1 Reserved, must be kept at reset value.

- **Bit 0 BYPASS**: power management unit bypass
  - 0: Power management unit normal operation. Use the internal regulator.
  - 1: Power management unit bypassed. Use the external power (voltage monitoring still active)

10.11.10 **PWR voltage monitor control register (PWR_VMCR)**

This register is protected against non-secure access when SCMSEC = 1 in the PWR_SECCFGR register.

This register is protected against unprivileged access when SCMSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when SCMSEC = 0 and NSPRIV = 1.

The PVDE and PLS bits are protected by lock mechanism. The lock control is in the SBS module controlled by PVDL bit in SBS_CFGR2 register. By default, the PVDE and PLS are unlocked.

Address offset: 0x034

Reset value: 0x0000 0000
10.11.11 PWR USB supply control register (PWR_USBSCR)

This register is protected against non-secure access when VUSBSEC = 1 in the PWR_SECCFG register.

This register is protected against unprivileged access when VUSBSEC = 1 and SPRIV = 1 in the PWR_PRIVCFG register, or when VUSBSEC = 0 and NSPRIV = 1.

Address offset: 0x038

Reset value: 0x0000 0000
### PWR voltage monitor status register (PWR_VMSR)

**Address offset:** 0x03C  
**Reset value:** 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value 1</th>
<th>Value 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>USB33RDY: VDDUSB ready</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>PVDO: programmable voltage detect output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>VDDIO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>AVDO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>2RDY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>AVDO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bit 24 USB33RDY: VDDUSB ready**  
- 0: VDDUSB is below the threshold of the VDDUSB voltage monitor.  
- 1: VDDUSB is equal or above the threshold of the VDDUSB voltage monitor.

**Bit 23 PVDO: programmable voltage detect output**  
- This bit is set and cleared by hardware. It is valid only if the PVD has been enabled by the PVDE bit.  
- 0: VDD is equal or higher than the PVD threshold selected through the PLS[2:0] bits.  
- 1: VDD is lower than the PVD threshold selected through the PLS[2:0] bits.  

**Note:** Since the PVD is disabled in Standby mode, this bit is equal to 0 after Standby or reset until the PVDE bit is set.

**Bit 21 Reserved, must be kept at reset value.**
Bit 20 **VDDIO2RDY**: voltage detector output on VDDIO2
   This bit is set and cleared by hardware.
   0: VDDIO2 is below 1.2 V.
   1: VDDIO2 is above or equal to 1.2 V.

Bit 19 **AVDO**: analog voltage detector output on VDDA
   This bit is set and cleared by hardware. It is valid only if AVD on VDDA is enabled by the
   AVDEN bit.
   0: VDDA is equal or higher than the AVD threshold selected with the ALS[2:0] bits.
   1: VDDA is lower than the AVD threshold selected with the ALS[2:0] bits.

   **Note**: Since the AVD is disabled in Standby mode, this bit is equal to 0 after standby or reset
   until the AVDEN bit is set.

   Bits 18:0 Reserved, must be kept at reset value.

### 10.11.13 PWR wake-up status clear register (PWR_WUSCR)

Each register bit CWUFx (x = 1 to 8) is protected against non-secure access when
WUPxSEC = 1 in the PWR_SECCFGR register.

Each bit CWUFx is protected against unprivileged access when WUPxSEC = 1 in
PWR_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when WUPxSEC 0 and
NSPRIV = 1.

Address offset: 0x040

Reset value: 0x0000 0000

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

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

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

Bits 7:0 **CWUFx**: clear wake-up pin flag for WUFx (x = 8 to 1)
   These bits are always read as 0.
   0: no effect
   1: writing 1 clears the WUFx wake-up pin flag (bit is cleared to 0 by hardware).

### 10.11.14 PWR wake-up status register (PWR_WUSR)

Address offset: 0x044

Reset value: 0x0000 0000
10.11.15 PWR wake-up configuration register (PWR_WUCR)

Each register bit WUPPUPDx (x = 1 to 8) and WUPPx (x = 1 to 8) and WUPENx (x = 1 to 8) is protected against non-secure access when WUPxSEC = 1 (x = 1 to 8) in the PWR_SECCFGR register.

Each bit WUPENx is protected against unprivileged access when WUPxSEC = 1 in PWR_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when WUPxSEC = 0 and NSPRIV=1.

Address offset: 0x048
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:16</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WUPPUPDx[1:0]</td>
<td>wake-up pin pull configuration for WKUPx (x = 8 to 1)</td>
<td>rw</td>
</tr>
<tr>
<td>WUPPx[1:0]</td>
<td>wake-up pin polarity bit for WUPx (x = 8 to 1)</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 WUPPUPDx[1:0]: wake-up pin pull configuration for WKUPx (x = 8 to 1)

These bits define the I/O pad pull configuration used when WUPENx = 1. The associated GPIO port pull configuration must be set to the same value or to 00. The wake-up pin pull configuration is kept in Standby mode.

00: no pull-up
01: pull-up
10: pull-down
11: reserved

Bits 15:8 WUPPx: wake-up pin polarity bit for WUPx (x = 8 to 1)

These bits define the polarity used for event detection on WUPx external wake-up pin.

0: detection on high level (rising edge)
1: detection on low level (falling edge)
Bits 7:0 **WUPENx**: enable wake-up pin WUPx (x = 8 to 1)
   These bits are set and cleared by software.
   0: an event on WUPx pin does not wake-up the system from Standby mode.
   1: a rising or falling edge on WUPx pin wakes up the system from Standby mode.
   
   **Note**: An additional wake-up event is detected if WUPx pin is enabled (by setting the WUPENx bit) when WUPx pin level is already high when WUPPx selects rising edge, or low when WUPPx selects falling edge.

### 10.11.16 PWR I/O retention register (PWR_IORETR)

This register is protected against non-secure access when RETSEC=1 in the PWR_SECCFGR register.

This register is protected against unprivileged access when RETBSEC=1 and SPRIV=1 in the PWR_PRIVCFGR register, or when RETSEC=0 and NSPRIV=1.

Address offset: 0x050

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>JTAGIORETEN: IO retention enable for JTAG IOs</td>
</tr>
<tr>
<td>20</td>
<td>0: IO Retention mode is disable.</td>
</tr>
<tr>
<td>19</td>
<td>1: IO Retention mode is enabling for PA13, PA14, PA15 and PB4.</td>
</tr>
<tr>
<td>18</td>
<td>when entering into standby mode, the output is sampled, and apply to the output IO during the standby power mode</td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>IORETEN: IO retention enable:</td>
</tr>
<tr>
<td>15</td>
<td>0: IO Retention mode is disabled.</td>
</tr>
<tr>
<td>14</td>
<td>1: IO Retention mode is enabling for all IO except the IO support the standby functionality and PA13, PA14, PA15 and PB4.</td>
</tr>
<tr>
<td>13</td>
<td>When entering into standby mode, the output is sampled, and apply to the output IO during the standby power mode</td>
</tr>
<tr>
<td>12</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>9</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>6</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>4</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>3</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>2</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>1</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

**Note**: The IO state is not retained if the DBG_STANDBY bit is set in DBGMCU_CR register.
### 10.11.17 PWR security configuration register (PWR_SECCFGR)

This register can be written only when the access is secure. It can be read by secure or non-secure access.

This register is write-protected against unprivileged write access when SPRIV=1 in the PWR_PRIVCFGR.

Address offset: 0x100
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

**Bit 15 VUSBSEC**: voltage USB secure protection
- 0: PWR_USBSCR can be read and written with secure or non-secure access.
- 1: PWR_USBSCR can be read and written only with secure access.

**Bit 14 VBSEC**: Backup domain secure protection
- 0: PWR_BDCR, PWR_DBPCR can be read and written with secure or non-secure access.
- 1: PWR_BDCR, PWR_DBPCR can be read and written only with secure access.

**Bit 13 SCMSEC**: supply configuration and monitoring secure protection.
- 0: PWR_SCCR and PWR_VMCR can be read and written with secure or non-secure access.
- 1: PWR_SCCR and PWR_VMCR can be read and written only with secure access.

**Bit 12 LPMSEC**: low-power modes secure protection
- 0: PWR_PMCR can be read and written with secure or non-secure access.
- 1: PWR_PMCR can be read and written only with secure access.

**Bit 11 RETSEC**: retention secure protection
- 0: PWR_IORETR can be read and written with secure or non-secure access.
- 1: PWR_IORETR can be read and written only with secure access.

Bits 10:8 Reserved, must be kept at reset value.

**Bits 7:0 WUPxSEC**: WUPx secure protection (x = 8 to 1)
- 0: The bits related to the WKUPx wake-up pin in PWR_WUSCR and PWR_WUCR can be read and written with secure or non-secure access.
- 1: The bits related to the WKUPx wake-up pin in PWR_WUSCR and PWR_WUCR can be read and written only with secure access.
10.11.18  **PWR privilege configuration register (PWR_PRIVCFGR)**

This register can be written only when the access is privileged. It can be read by privileged or unprivileged access.

**Address offset:** 0x104

**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-2</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td><strong>NSPRIV</strong>: PWR non-secure functions privilege configuration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and reset by software. This bit can be written only by privileged access, secure or non-secure.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Read and write to PWR non-secure functions can be done by privileged or unprivileged access.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Read and write to PWR non-secure functions can be done by privileged access only.</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td><strong>SPRIV</strong>: PWR secure functions privilege configuration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and reset by software. This bit can be written only by a secure privileged access.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Read and write to PWR secure functions can be done by privileged or unprivileged access.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Read and write to PWR secure functions can be done by privileged access only.</td>
<td></td>
</tr>
</tbody>
</table>
## 10.11.19 PWR register map

**Table 100. PWR register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>PWR_PMCR</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x04</td>
<td>PWR_PMSR</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x08 to 0x0FF</td>
<td>Reserved</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x10</td>
<td>PWR_VOSSR</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x14</td>
<td>PWR_VOSCR</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x18 to 0x1F</td>
<td>Reserved</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x20</td>
<td>PWR_BDCR</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x24</td>
<td>PWR_DBPCR</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x28</td>
<td>PWR_BDRS</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x2C</td>
<td>PWR_UCPDR</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x30</td>
<td>PWR_SCCR</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x34</td>
<td>PWR_VMCR</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x38</td>
<td>PWR_USBSCR</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Refer to Section 2.3 for the register boundary addresses.
11 

Reset and clock control (RCC)

11.1 Introduction

The reset and clock control (RCC) manages the different resets, and generates the clocks for the bus and peripherals.

11.2 RCC pins and internal signals

The table below lists the RCC inputs and output signals connected to package pins or balls.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRST</td>
<td>I/O</td>
<td>System reset, can be used to provide reset to external devices</td>
</tr>
<tr>
<td>OSC32_IN</td>
<td>I</td>
<td>32 kHz oscillator input</td>
</tr>
<tr>
<td>OSC32_OUT</td>
<td>O</td>
<td>32 kHz oscillator output</td>
</tr>
<tr>
<td>OSC_IN</td>
<td>I</td>
<td>System oscillator input</td>
</tr>
<tr>
<td>OSC_OUT</td>
<td>O</td>
<td>System oscillator output</td>
</tr>
<tr>
<td>MCO</td>
<td>O</td>
<td>Output clock for external devices</td>
</tr>
<tr>
<td>LSCO</td>
<td>O</td>
<td>Low-speed output clock for external devices</td>
</tr>
<tr>
<td>AUDIOCLK</td>
<td>I</td>
<td>External kernel clock input for SAI1, SAI2, I2S1, I2S2, and I2S3</td>
</tr>
</tbody>
</table>

11.3 RCC reset functional description

There are three types of reset:
- a system reset
- a power reset
- a Backup domain reset

11.3.1 Power reset

A power reset is generated when one of the following events occurs:
- a brownout reset (BOR)
- when exiting Standby mode

A brownout reset, including power-on or power-down reset (POR/PDR), sets all registers to their reset values except the ones in the Backup domain.

When exiting Standby mode, all registers in the core domain are set to their reset value. Registers outside the core domain (RTC, WKUP, IWDG, and GPIO pullup/pulldown configuration during Standby and Standby mode exit) are not impacted.
11.3.2 System reset

A system reset sets all registers to their reset values except the reset flags in **RCC reset status register (RCC_RSR)** and the registers in the Backup domain.

A system reset is generated when one of the following events occurs:

- a low level on the NRST pin (external reset)
- a window watchdog event (WWDG reset)
- an independent watchdog event (IWDG reset)
- a software reset (SW reset) (see **Software reset**)
- a low-power mode security reset (see **Low-power mode security reset**)
- a brownout reset

The reset source can be identified by checking the reset flags in **RCC reset status register (RCC_RSR)**.

These sources act on the NRST pin and this pin is always kept low during the delay phase. The reset service routine vector is selected depending on product state, on Boot option bytes or on both.

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source. In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

In case on an internal reset, the internal pull-up R PU is deactivated to save the power consumption through the pull-up resistor.

---

**Figure 47. Simplified diagram of the reset circuit**

![Diagram of the reset circuit](image-url)

Software reset

The SYSRESETREQ bit in Cortex-M33 application interrupt and reset control register must be set to force a software reset on the device.
Low-power mode security reset

To avoid that critical applications mistakenly enter a low-power mode, the following low-power mode security resets are available. If enabled in option bytes, the resets are generated in any of the following conditions:

- Entering Standby mode: this type of reset is enabled by resetting nRST_STDBY bit in user option bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.
- Entering Stop mode: this type of reset is enabled by resetting nRST_STOP bit in user option bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode.

For further information on the user option bytes, refer to Section 7.4.1: Option bytes description.

11.3.3 Backup domain reset

The Backup domain has two specific resets, generated when one of the following events occurs:

- a software reset, triggered by setting the VSWRST bit in the RCC Backup domain control register (RCC_BDCR). Write access to this domain must be enabled before setting VSWRST bit to perform the reset.
- a VDD or VBAT power on, if both supplies have previously been powered off

A Backup domain reset affects the LSE oscillator, the RTC, the backup registers, the backup SRAM, and the RCC_BDCR register.

11.3.4 Reset source identification

The application can identify the reset source by checking the reset flags in the RCC_RSR register.

The software can reset the flags by setting RMVF bit.

Table 102 shows how the status bits of the RCC_RSR register behave according to the situation that generated the reset. For example, when an IWDG timeout occurs, if the CPU is reading the RCC_RSR register during the boot phase, both PINRSTF and IWDGRSTF bits are set, indicating that the IWDG also generated a pin reset.

<table>
<thead>
<tr>
<th>Reset</th>
<th>LPWRRSTF</th>
<th>WWDRSTF</th>
<th>IWDGRSTF</th>
<th>SFTRSTF</th>
<th>BORRSTF</th>
<th>PINRSTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  Power-on reset</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2  Pin/pad reset</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3  Brownout (low or high) reset</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4  System reset generated by CPU</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5  WWDG reset</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
11.4 RCC clocks functional description

Four different clock sources can be used to drive the system clock (SYSCLK):
- HSI: high-speed internal up to 64 MHz RC oscillator clock
- CSI: low power internal RC oscillator clock
- HSE: high-speed external crystal or clock, from 4 to 50 MHz
- PLL1 clock

The HSI is used as system clock source after startup from reset, configured at 32 MHz.

The device has the following additional clock sources:
- LSI: 32 kHz low-speed internal RC that drives the independent watchdog and optionally the RTC used for auto-wake-up from Stop and Standby modes
- LSE: 32.768 kHz low-speed external crystal or clock that optionally drives the real-time clock (rtc_ck)
- HSI48: internal 48 MHz RC that potentially drives the USB FS and the RNG
- PLL2 and PLL3 clocks

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

Several prescalers can be used to configure the AHB frequency, the APB1 and APB2 domains. The maximum frequency of the AHB and APB domains is 250 MHz.

All the peripheral clocks are derived from their bus clock (HCLK, PCLK1, PCLK2 or PCLK3) except the following ones that receive an independent kernel clock. This kernel clock can be selected by software between several sources thanks to RCC_CCIPRx registers (x = 1,2,3, 4,5).

In addition, the RTC kernel clock is selected by software in RCC_BDCR. The IWDG clock is always the LSI 32 kHz clock.

The RCC feeds the Cortex system timer (SysTick) external clock with the AHB clock (HCLK) divided by eight, or LSE or LSI. The SysTick can work either with this clock or directly with the Cortex clock (HCLK), configurable in the SysTick control and status register.

FCLK acts as Cortex-M33 free-running clock.
Figure 48. Clock tree

1. For details on the internal/external clock source, refer to the Electrical characteristics in the datasheet.
11.4.1 HSE clock

The HSE block can generate a clock from two possible sources:
- an external crystal/ceramic resonator
- an external clock source

External clock source (HSE bypass)

In this mode, an external clock source must be provided to OSC_IN pin. The external clock can be low swing (analog) or digital. If this clock is directly used by a peripheral, the duty cycle requirement is defined by the peripheral and the application (refer to datasheet for more details).

In case of an analog clock (low swing) the HSEBYP and HSEON bits must be set to 1 in the RCC clock control register (RCC_CR).

In case of a digital clock, the HSEBYP and the HSEEXT bits must be set to 1 followed by setting the HSEON bit to 1 in the RCC clock control register (RCC_CR).

External crystal/ceramic resonator

The oscillator is enabled by setting the HSEBYP bit to 0 and HSEON bit to 1.

The HSE can be used when the product requires a very accurate high-speed clock.

The associated hardware configuration is shown in Figure 49: the resonator and the load capacitors must be placed as close as possible to the oscillator pins to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The HSERDY flag of the RCC clock control register (RCC_CR) indicates whether the HSE oscillator is stable or not. At startup, the hse_ck clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the RCC clock source interrupt enable register (RCC_CIER).
The HSE can be switched ON and OFF through the HSEON bit. Note that the HSE cannot be switched OFF if one of the following two conditions is met:
- the HSE is used directly (via software mux) as system clock
- the HSE is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the HSEON bit to 0.

The HSE is automatically disabled by hardware, when the system enters Stop or Standby mode.

In addition, the HSE clock can be driven to the MCO1 and MCO2 outputs and used as clock source for other application components.

11.4.2 HSI clock

The HSI block provides the default clock to the product.

The HSI is a high-speed internal RC oscillator that can be used directly as system clock, peripheral clock, or as PLL input. A predivider allows the application to select an HSI output frequency of 8, 16, 32 or 64 MHz. This predivider is controlled by the HSIDIV.

The HSI advantages are the following:
- low-cost clock source, as no external crystal is required
- faster startup time than HSE (a few microseconds)

The HSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

The HSI can be switched ON and OFF using the HSION bit. Note that the HSI cannot be switched OFF if one of the two conditions is met:
- the HSI is used directly (via software mux) as system clock
- the HSI is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the HSION bit to 0. Note that the HSIDIV cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set to 1). In that case the hardware does not update the HSIDIV with the new value. However it is possible to change the HSIDIV if the HSI is used directly as system clock.

The HSIRDY flag indicates if the HSI is stable or not. At startup, the HSI output clock is not released until this bit is set by hardware.

The HSI clock can also be used as a backup source (auxiliary clock) if the HSE fails (refer to Section 11.4.10: Clock security system (CSS)). The HSI can be disabled or not when the system enters Stop mode.

In addition, the HSI clock can be driven to the MCO1 output and used as clock source for other application components.

Care must be taken when the HSI is used as kernel clock for communication peripherals, the application must take into account the following parameters:
- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available
- the frequency accuracy.
Note: The HSI can remain enabled when the system is in Stop mode.

HSI calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. That is why each device is factory calibrated by STMicroelectronics to achieve an accuracy of ACCHSI (refer to the product datasheet for more information).

After a power-on reset, the factory calibration value is loaded in the HSICAL[11:0] bits. If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user application can trim the HSI frequency using the HSITRIM[6:0] bits.

Note: HSICAL[11:0] and HSITRIM[6:0] bits are located in the RCC CSI calibration register (RCC_CSICFGR).

11.4.3 CSI oscillator

The CSI is a low-power RC oscillator that can be used directly as system clock, peripheral clock, or PLL input.

The CSI advantages are the following:

- low-cost clock source since no external crystal is required
- faster startup time than HSE (a few microseconds)
- very low-power consumption,

The CSI provides a clock frequency of about 4 MHz, while the HSI is able to provide a clock up to 64 MHz.

CSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

The CSI can be switched ON and OFF through the CSION bit. The CSIRDY flag indicates whether the CSI is stable or not. At startup, the CSI output clock is not released until this bit is set by hardware.

The CSI cannot be switched OFF if one of the two conditions is met:

- The CSI is used directly (via software mux) as system clock.
- The CSI is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the CSION bit to 0.

The CSI can be disabled or not when the system enters Stop mode.

In addition, the CSI clock can be driven to the MCO2 output and used as clock source for other application components.

Even if the CSI settling time is faster than the HSI, care must be taken when the CSI is used as kernel clock for communication peripherals: the application must take into account the following parameters:

- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available,
- the frequency precision.

Note: CSION and CSIRDY bits are located in the RCC clock control register (RCC_CR).
CSI calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated to achieve ACCCSI accuracy (refer to the product datasheet for more information). After reset, the factory calibration value is loaded in the CSICAL[7:0] bits.

Voltage and/or temperature variations affect the RC oscillator frequency. The user application can trim the CSI frequency using the CSITRIM[5:0] bits.

**Note:** Bits CSICAL[7:0] and CSITRIM[5:0] are located into the RCC CSI calibration register (RCC_CSICFRG).

![Figure 50. CSI calibration flow](image)

11.4.4 HSI48 clock

The HSI48 clock signal is generated from an internal 48 MHz RC oscillator, and can be used directly for USB and for random number generator (RNG).

The internal 48 MHz RC oscillator is mainly dedicated to provide a high-precision clock to the USB peripheral by means of a special clock recovery system (CRS) circuitry. The CRS can use the USB SOF signal, the LSE, or an external signal to automatically and quickly adjust the oscillator frequency on-the-fly. It is disabled as soon as the system enters Stop or Standby mode. When the CRS is not used, the HSI48 RC oscillator runs on its default frequency, subject to manufacturing process variations.

For more details on how to configure and use the CRS peripheral, refer to Section 12: Clock recovery system (CRS).

The HSI48RDY flag in the RCC_CR register indicates whether the HSI48 RC oscillator is stable or not. At startup, the HSI48 RC oscillator output clock is not released until this bit is set by hardware.

The HSI48 can be switched on and off using the HSI48ON bit in the RCC_CR register.

11.4.5 PLL description

The RCC features three PLLs:
- a main PLL (PLL1), generally used to provide clocks to the CPU and some peripherals
- two dedicated PLLs (PLL2 and PLL3), used to generate the kernel clock for peripherals
The PLLs integrated into the RCC are completely independent. They offer the following features:

- A VCO supporting two modes:
  - a wide-range
  - a low-range used, for instance, in audio applications
- Input frequency range:
  - 2 to 16 MHz for the VCO in wide-range mode
  - 1 to 2 MHz for the VCO in low-range mode
- Capability to work either in integer or fractional mode
- 13-bit sigma-delta modulator, allowing to fine-tune the VCO frequency by steps of 11 to 0.3 ppm
- The sigma-delta modulator can be updated on-the-fly without generating frequency overshoots on PLLs outputs
- Each PLL offers three outputs with post-dividers

**Figure 51. PLL block diagram**

The PLLs are controlled via RCC_PLLxDIVR, RCC_PLLxFRACR, RCC_PLLCFGR, and RCC_CR registers.

The frequency of the reference clock provided to the PLLs (refx_ck) must range from 1 to 16 MHz. The PLLxM dividers of the RCC PLLx clock source selection register (RCC_PLLxCFGR) must be properly programmed to match this condition. In addition, the PLLxRGE[1:0] field of the RCC PLLx clock source selection register (RCC_PLLxCFGR) must be set according to the reference input frequency to optimize performance.

The user application can then configure the VCO. The smaller range (150 to 420 MHz) must be chosen when the reference clock frequency is lower than 2 MHz.

To reduce the power consumption, it is recommended to configure the VCO output to the smaller range.

DIVNx loop divider must be programmed to achieve the expected frequency at VCO output. In addition, the VCO output range must be respected.

The PLLs operate in integer mode when the value of SH_REG bit of the FRACNx shadow register is set to 0. The SH_REG bit is updated with the FRACNx value when PLLxFRACEN
bit goes from 0 to 1. The sigma-delta modulator is designed to minimize the jitter impact, while allowing very small frequency steps.

The PLLs can be enabled by setting PLLxON to 1. The PLLxRDY bits indicate that the PLL is ready (locked).

Note: Before enabling the PLLs, make sure that the reference frequency \((\text{refx}_\text{ck})\) provided to it is stable, so the hardware does not allow changing PLLxM when the PLLx is ON, and it is also not possible to change PLLSRC when one of the PLLs is ON.

The hardware prevents writing PLL1ON to 0 if the PLL1 is currently used to deliver the system clock. There are other hardware protections on the clock generators (refer to HSE clock, HSI clock, and CSI oscillator).

The following PLL parameters cannot be changed once the PLL is enabled: DIVNx, PLLxRGE, PLLxVCOSEL, PLLxP (DIVP), PLLxQ (DIVQ), and PLLxR (DIVR).

For optimal behavior of the PLL when one of the post-divider (DIVP, DIVQ or DIVR) is not used, the application must set the enable bit (DIVyEN) as well as the corresponding post-divider bits (DIVP, DIVQ or DIVR) to 0.

If the above rules are not respected, the PLL output frequency is not guaranteed.

### Output frequency computation

When the PLL is configured in integer mode \((\text{SH}\_\text{REG} = 0)\), the VCO frequency \((\text{F}_{\text{VCO}})\) is given by the following expression:

\[
\text{F}_{\text{VCO}} = \text{F}_{\text{REF}_\text{CK}} \times \text{DIVN}
\]

When the PLL is configured in fractional mode \((\text{SH}\_\text{REG} \neq 0)\), the DIVN divider must be initialized before enabling the PLLs. However, it is possible to change the value of FRACNx on-the-fly without disturbing the PLL output.

This feature can be used either to generate a specific frequency from any crystal value with a good accuracy, or to fine-tune the frequency on-the-fly.

For each PLL, the VCO frequency is given by the following formula:

\[
\text{F}_{\text{VCO}} = \text{F}_{\text{ref}_\text{ck}} \times \left(\frac{\text{DIVN} + \text{FRACN}}{2^{13}}\right)
\]

Note: For PLL1, DIVP can only take odd values.

The PLLs are disabled by hardware when:
- the system enters Stop or Standby mode
- an HSE failure occurs when HSE or PLL (clocked by HSE) are used as system clock

### PLL initialization phase

Figure 52 shows the recommended PLL initialization sequence in integer and fractional mode. The PLLx are supposed to be disabled at the start of the initialization sequence:
1. Initialize the PLLs registers according to the required frequency.
   - Set PLLxFRACEN of RCC PLLx clock source selection register (RCC_PLLxCFGR) to 0 for integer mode.
   - For fractional mode, set FRACN to the required initial value (FracInitValue) and then set PLLxFRACEN to 1.
2. Once the PLLxON bit is set to 1, the user application must wait until PLLxRDY bit is set to 1. If the PLLx is in fractional mode, the PLLxFRACEN bit must not be set back to 0 as long as PLLxRDY = 0.
3. Once the PLLxRDY bit is set to 1, the PLLx is ready to be used.
4. If the application intends to tune the PLLx frequency on-the-fly (possible only in fractional mode), then:
   a) PLLxFRACEN must be set to 0. When PLLxFRACEN = 0, the sigma-delta modulator is still operating with the value latched into SH_REG. The application must wait for three clock periods of refx_ck (PLLxFRACEN bit propagation delay).
   b) A new value must be uploaded into PLLxFRACR (FracValue(n)).
   c) PLLxFRACEN must be set to 1, to latch the content of PLLxFRACR into its shadow register. The new value is considered after three clock periods of refx_ck (PLLxFRACEN bit propagation delay).

**Note:** When the PLLxRDY goes to 1 the difference between the PLLx output frequency and the target value is lower than ±2%. 
### 11.4.6 LSE clock

The LSE block can generate a clock from two possible sources:
- an external crystal/ceramic resonator
- an external user clock

**External clock source (LSE bypass)**

In this mode, an external clock source must be provided to OSC32_IN pin. The input clock can have a frequency up to 1 MHz, and be low swing (analog) or digital. A duty cycle close to 50% is recommended.

This external clock is provided to the OSC32_IN pin while the OSC32_OUT pin must be left Hi-Z (see Figure 48).
In case of an analog clock (low swing), the LSEBYP and LSEON bits must be set to 1 (RCC Backup domain control register (RCC_BDCR)).

In case of a digital clock, the LSEBYP and the LSEEXT bits must be set to 1 followed by setting the LSEON bit to 1 (RCC Backup domain control register (RCC_BDCR)). If the RTC is used, the LSE bypass must not be configured in digital mode, but in low swing analog mode (default value after reset).

External crystal/ceramic resonator (LSE crystal)

The LSE clock is generated from a 32.768 kHz crystal or ceramic resonator. It has the advantage to provide a low-power, highly accurate clock source to the real-time clock (RTC) for clock/calendar or other timing functions.

The LSERDY flag of the RCC Backup domain control register (RCC_BDCR) indicates whether the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in the RCC clock source interrupt enable register (RCC_CIER).

The LSE oscillator is switched ON and OFF using the LSEON bit. The LSE remains enabled when the system enters Stop or Standby mode.

In addition, the LSE clock can be driven to the MCO1 output and used as clock source for other application components.

The LSE also offers a programmable driving capability (LSEDRV[1:0]), which can be used to modulate the amplifier driving capability. This driving capability is chosen according to the external crystal/ceramic component requirement to ensure a stable oscillation.

The driving capability must be set before enabling the LSE oscillator.

11.4.7 LSI clock

The LSI acts as a low-power clock source that can be kept running when the system is in Stop or Standby mode for the independent watchdog (IWDG) and auto-wakeup unit (AWU).

The clock frequency is around 32 kHz. For more details, refer to the electrical characteristics section of the datasheet.

The LSI can be switched ON and OFF using the LSION bit. The LSIRDY flag indicates whether the LSI oscillator is stable or not. If an independent watchdog is started either by hardware or software, the LSI is forced ON and cannot be disabled.

The LSI remains enabled when the system enters Stop or Standby mode.

At LSI startup, the clock is not provided until the hardware sets the LSIRDY bit. An interrupt can be generated if enabled in the RCC clock source interrupt enable register (RCC_CIER).

In addition, the LSI clock can be driven to the MCO2 output and used as a clock source for other application components.

Note: Bits LSION and LSIRDY are located in the RCC Backup domain control register (RCC_BDCR).
11.4.8 System clock (SYSCLK) selection

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI oscillator
- CSI oscillator
- HSE oscillator
- PLL

The system clock maximum frequency is 250 MHz. After a system reset (or after leaving Standby mode), the HSI oscillator, at 32 MHz, is selected as system clock. When a clock source is used directly or through the PLL as a system clock, it is not possible to stop it. A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source not yet ready is selected, the switch occurs when the clock source becomes ready. Status bits in the RCC clock control register (RCC_CR) indicate which clocks are ready, and which clock is currently used as a system clock.

11.4.9 Handling clock generators in stop and standby modes

When the whole system enters Stop mode, all the clocks (system and kernel clocks) are stopped, as well as the following clock sources:

- CSI, HSI (depending on HSIKERON and CSIKERON bits)
- HSE
- PLL1 and PLL2
- HSI48

The content of the RCC registers is not altered except for PLL1ON, PLL2ON, HSEON, and HSI48ON, set to 0.

Exiting Stop mode

When the system exits this mode via a wake-up event, the application can select which oscillator (HSI and/or CSI) is used to restart. The STOPWUCK bit selects the oscillator used as system clock. The STOPKERWUCK bit selects the oscillator used as kernel clock for peripherals. The STOPKERWUCK bit is useful if after a system Stop, a peripheral needs a kernel clock generated by an oscillator different from the one used for the system clock.

These bits belong to the RCC clock configuration register1 (RCC_CFGR1)

<table>
<thead>
<tr>
<th>STOPWUCK</th>
<th>STOPKERWUCK</th>
<th>Activated oscillator when the system exits Stop mode</th>
<th>Distributed clocks when the system exits Stop mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>System clock</td>
<td>Kernel clock</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>HSI</td>
<td>HSI</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>HSI and CSI</td>
<td>HSI and/or CSI</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>CSI</td>
<td>CSI</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>CSI</td>
<td>CSI</td>
</tr>
</tbody>
</table>

Table 103. STOPWUCK and STOPKERWUCK description
During Stop mode

There are two specific cases where the HSI or CSI can be enabled during this mode.

- When a dedicated peripheral requests the kernel clock the peripheral receives the HSI or CSI according to the kernel clock source selected for this peripheral (via CKPERSEL[1:0]).
- When the HSikerON or CSikerON bits of the **RCC clock control register (RCC_CR)** are set, the HSI and CSI are kept running, but the outputs are gated. The clock is then available immediately when the system exits Stop mode, or when a peripheral requests the kernel clock (see Table 101 for details).

<table>
<thead>
<tr>
<th>HSikerON (CSikerON)</th>
<th>HSI (CSI) state during Stop mode</th>
<th>HSI (CSI) state setting time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OFF</td>
<td>$t_{su(HSI)}$, $t_{su(CSI)}^{(1)}$</td>
</tr>
<tr>
<td>1</td>
<td>Running and gated</td>
<td>Immediate</td>
</tr>
</tbody>
</table>

1. $t_{su(HSI)}$ and $t_{su(CSI)}$ are the startup times of, respectively, the HSI and CSI oscillators (refer to the product datasheet for the values of these parameters).

When the microcontroller exists system standby mode, the HSI is selected as system and kernel clock. The RCC registers are reset to their initial values except for the RCC_RSR and RCC_BDCR registers.

Note: The HSI and CSI outputs provide two clock paths:

- one path for the system clock (hsi_ck or csi_ck)
- one path for the peripheral kernel clock (hsi_ker_ck or csi_ker_ck).

When a peripheral requests the kernel clock in system stop mode, only the path providing the hsi_ker_ck or csi_ker_ck is activated.

11.4.10 Clock security system (CSS)

Clock security system on HSE

The clock security system can be enabled by software via the HSECSSON bit, which can be enabled even when the HSEON is set to 0.

The CSS on HSE is enabled by the hardware when the HSE is enabled and ready, and HSECSSON set to 1.

The CSS on HSE is disabled when the HSE is disabled. As a result, this function does not work when the system is in Stop mode.

It is not possible to clear directly the HSECSSON bit by software.

The HSECSSON bit is cleared by hardware when a system reset occurs or when the system enters Standby mode.

If a failure is detected on the HSE clock, the system automatically switches to the HSI or CSI, depending on STOPWUCK bit configuration in **RCC clock configuration register1 (RCC_CFGR1)**, to provide a safe clock. The HSE is then automatically disabled, a clock failure event is sent to the break inputs of advanced-control timer (TIM1), and an NMI is automatically generated to inform the application about the failure, allowing the MCU to
perform rescue operations. If the HSE output was used as clock source for PLLs when the failure occurred, the PLLs are also disabled.

If an HSE clock failure occurs when the CSS is enabled, the CSS generates an interrupt that causes the automatic generation of an NMI. The HSECSSF flag in **RCC clock source interrupt flag register (RCC_CIFR)** is set to 1 to allow the application to identify the failure source. The NMI routine is executed indefinitely until the HSECSSF bit is cleared. As a consequence, the application must clear the HSECSSF flag in the NMI ISR by setting the HSECSSC bit in the **RCC clock source interrupt clear register (RCC_CICR)**.

### Clock security system on LSE

A clock security system on LSE can be activated by software writing the LSECSSON bit in the **RCC Backup domain control register (RCC_BDCR)**. This bit can be disabled only by a hardware reset or RTC software reset, or after a failure detection on LSE. LSECSSON must be written after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware), and after the RTC clock has been selected by RTCSEL.

The CSS on LSE is working in all modes including VBAT. It works also under system reset (excluding power-on reset).

The clock security system on LSE detects when the LSE disappears or in case of over frequency. In addition, the glitches on LSE can be filtered by setting LSEGFON. LSEGFON must be written when the LSE is disabled (LSEON = 0 and LSERDY = 0).

If a failure is detected on the external 32 kHz oscillator, the LSE clock is no longer supplied to the RTC, but no hardware action is made to the registers.

The CSS on LSE detection event is connected to the internal tamper 3 of the TAMP peripheral. The internal tamper 3 must be enabled (ITAMP3E = 1 in TAMP_CR1 register) and the associated interrupt enabled (ITAMP3IE in TAMP_IER) to wake up from the low-power modes. This erases also the TAMP backup registers and backup SRAM unless the ITAMP3NOER = 1 in the TAMP_CR3 (see **Section 47: Tamper and backup registers (TAMP)** for more details).

In case of CSS on LSE detection event (LSECSSD = 1 in the RCC_BDCR), the software must disable the LSECSSON bit, stop the defective 32 kHz oscillator (disabling LSEON), and change the RTC clock source (no clock or LSI or HSE, with RTCSEL), or take actions to secure the application.

Refer to datasheet for CSS on LSE electrical characteristics.

#### 11.4.11 Clock output generation (MCO1/MCO2)

Two microcontroller clock output pins (MCO1 and MCO2) are available. A clock source can be selected for each output. The selected clock can be divided thanks to configurable prescaler (refer to **Figure 48** for additional information on signal selection).

MCO1 and MCO2 outputs are controlled via MCO1PRE[3:0], MCO1[2:0], MCO2PRE[3:0], and MCO2[2:0], located in the **RCC clock configuration register1 (RCC_CFRG1)**.

The GPIO port corresponding to each MCO pin must be programmed in alternate function mode.

The clock provided to the MCOs outputs must not exceed the maximum pin speed (refer to the product datasheet for information on the supported speed).
Another output (LSCO) allows one of the low-speed clocks (LSI, LSE) to be output onto the external LSCO pin. This output is available in Stop mode, not available in Standby and VBAT modes. The selection is controlled by the LSCOSEL bit, and enabled by the LSCOEN bit in the **RCC Backup domain control register (RCC_BDCR)**.

The MCO clock output requires the corresponding alternate function selected on the MCO pin. The LSCO pin must be left in default POR state.

### 11.4.12 Kernel clock selection

Some peripherals are designed to work with two different clock domains that operate asynchronously:

- a clock domain synchronous with the register and bus interface (**ckg_bus_perx** clock)
- a clock domain generally synchronous with the peripheral (kernel clock)

The benefit of having peripherals supporting these two clock domains is that the user application has more freedom to choose optimized clock frequency for the CPU, bus matrix and for the kernel part of the peripheral. The user application can thus change the bus frequency without reprogramming the peripherals. As an example, an ongoing transfer with UART is not disturbed if its APB clock is changed on-the-fly.

*Table 105* shows the kernel clock that the RCC can deliver to the peripherals. Each row represents a multiplexer and the peripherals connected to its output.
To reduce the number of switches, some peripherals share the same kernel clock source. Nevertheless, all peripherals have their dedicated enable signal.

Table 105. Kernel clock distribution overview (continued)

<table>
<thead>
<tr>
<th>Peripherals</th>
<th>Clock multiplexer control bits</th>
<th>pll1_q_ck</th>
<th>pll2_p_ck</th>
<th>pll2_q_ck</th>
<th>pll3_p_ck</th>
<th>pll3_q_ck</th>
<th>pll3_r_ck</th>
<th>sys_ck</th>
<th>hse_ck</th>
<th>hsi_ker_ck</th>
<th>csi_ker_ck</th>
<th>hsi48_ck</th>
<th>lse_ck</th>
<th>lsi_ck</th>
<th>per_ck</th>
<th>AUDIOCLK</th>
<th>Disabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM[8:1]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIM[17:12]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIM16/17</td>
<td>TIMICSEL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>RNG</td>
<td>RNGSEL</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SAI1</td>
<td>SAI1SEL</td>
<td>0 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4 3</td>
<td>-</td>
</tr>
<tr>
<td>SAI2</td>
<td>SAI2SEL</td>
<td>0 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4 3</td>
<td>-</td>
</tr>
<tr>
<td>SPI(I2S)1</td>
<td>SPI1SEL</td>
<td>0 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4 3</td>
<td>-</td>
</tr>
<tr>
<td>SPI(I2S)2</td>
<td>SPI2SEL</td>
<td>0 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4 3</td>
<td>-</td>
</tr>
<tr>
<td>SPI(I2S)3</td>
<td>SPI3SEL</td>
<td>0 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4 3</td>
<td>-</td>
</tr>
<tr>
<td>SPI4</td>
<td>SPI4SEL</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>0 5 3 4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SPI5</td>
<td>SPI5SEL</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>0 5 3 4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SPI6</td>
<td>SPI6SEL</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>0 5 3 4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>USARTx</td>
<td>USARTxSEL</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>0 3 4 5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LPUART1</td>
<td>LPUART1SEL</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>0 3 4 5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>USB</td>
<td>USBSSEL</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>ADCDAC(5)</td>
<td>ADCDACSEL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0 3 4 5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DAC</td>
<td>DACSEL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UCPD1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x(6)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ETH (ptp)</td>
<td>-</td>
<td>x</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RTC/AWU</td>
<td>RTCSEL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3(7)</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>All</td>
<td>CKPERSEL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2 0 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>-</td>
</tr>
</tbody>
</table>

1. The bus (AP or AHB) clocks are the bus interface clocks to whom the peripherals are connected.
2. The per_ck clock can be hse_ck, hsi_ker_ck, or csi_ker_ck, according to CKPERSEL selection.
3. With a duty cycle close to 50%, meaning that DIV[P/Q/R]x, values must be even. For SDMMCx, the duty cycle must be 50% when supporting DDR.
4. Clock CSI divided by 122.
5. With a duty cycle close to 50%, meaning that DIV[P/Q/R]x, values must be even. For ADC, the duty cycle must be 50% when supporting DDR.
7. Clock HSE divided by RTCPRE.
Peripherals dedicated to audio applications

The audio peripherals generally need specific accurate frequencies, the kernel clock of the SAIs or SPI/I2Ss can be generated by:

- PLL1 when the amount of active PLLs must be reduced (for SAIs and SPI/I2S1 to 3)
- APB2 peripheral clock (for SPI/I2S4 and 5)
- APB3 peripheral clock (for SPI/I2S6)
- PLL2 or 3 for optimal flexibility in frequency generation
- HSE, HSI or CSI for use-cases where the current consumption is critical
- AUDIOCLK when an external clock reference needs to be used

Peripherals dedicated to control and data transfer

Peripherals such as SPIs, I2Cs, UARTs do not need a specific kernel clock frequency but a clock fast enough to generate the correct baud rate, or the required bit clock on the serial interface. For that purpose the source can be selected among the following ones:

- PLL1 when the amount of active PLLs must be reduced
- PLL2 or PLL3 if better flexibility is required. As an example, this solution allows changing the frequency bus via PLL1 without affecting the speed of some serial interfaces.
- HSI or CSI for low-power use-cases or when the peripheral must quickly wake up from Stop mode (such as UART or I2C)

Note: *UARTs also need the LSE clock when high baud rates are not required.*

The OCTOSPI and SDMMC1/2 can also use a clock different from the bus interface one for more flexibility.

RTC/AWU clock

The rtc_clk clock source can be one of the following:

- the hse_1M_ck (hse_ck divided by a programmable prescaler)
- the lse_ck
- the lsi_ck clock

The source clock is selected by programming the RTCSEL[1:0] bits in the **RCC Backup** domain control register (**RCC_BDCR**) and the RTCPRE[5:0] bits in the **RCC clock configuration register1** (**RCC_CFRG1**).

This selection cannot be modified without resetting the Backup domain.

If the LSE is selected as RTC clock, the RTC works normally even if the backup or the V_DD supply disappears.

The LSE clock is in the Backup domain, whereas the other oscillators are not. As a consequence:

- If LSE is selected as RTC clock, the RTC continues working even if the V_DD supply is switched OFF, provided the V_BAT supply is maintained.
- If LSI is selected as the RTC clock, the AWU state is not guaranteed if the V_DD supply is powered off.
- If the HSE clock is used as RTC clock, the RTC state is not guaranteed if the V_DD supply is powered off or if the V_CORE supply is powered off.
The \texttt{rtc\_ck} clock is enabled through RTCEN bit located in the \textit{RCC Backup domain control register (RCC\_BDCR)}.

The RTC bus interface clock (APB clock) is enabled through RTCAPBEN and RTCAPBLEN bits located in RCC\_APB3ENR/LPENR registers.

\textbf{Note:} To read the RTC calendar register when the APB clock frequency is less than seven times the RTC clock frequency ($F_{APB} < 7 \times F_{RTC\text{CLK}}$), the software must read the calendar time and date registers twice. The data are correct if the second read access to RTC\_TR gives the same result than the first one. Otherwise a third read access must be performed.

\section*{Watchdog clocks}

The RCC provides the clock for the two watchdog blocks available on the circuit. The independent watchdog (IWDG) is connected to the LSI. The window watchdog (WWDG) is connected to the APB clock.

If an independent watchdog is started by either hardware option or software access, the LSI is forced ON and cannot be disabled. After the LSI oscillator setup delay, the clock is provided to the IWDG.

\section*{Clock frequency measurement using TIMx}

Most of the clock source generator frequencies can be measured by means of the input capture of TIMx.

- Calibrating the HSI or CSI with the LSE:
  The primary purpose of having the LSE connected to a TIMx input capture is to be able to accurately measure the HSI or CSI. This requires to use the HSI or CSI as system clock source either directly or via PLL1. The number of system clock counts between consecutive edges of the LSE signal gives a measurement of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm) we can determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process and/or temperature- and voltage-related frequency deviations.

  The basic concept consists in providing a relative measurement (e.g. HSI/LSE ratio). The precision is therefore tightly linked to the ratio between the two clock sources. The greater the ratio is, the more accurate the measurement is.

  The HSI and CSI oscillators have dedicated user-accessible calibration bits for this purpose (see \textit{RCC CSI calibration register (RCC\_CSICFGR)}). When HSI or CSI is used via the PLLx, the system clock can also be fine-tuned by using the fractional divider of the PLLs.

- Calibrating the LSI with the HSI:
  The LSI frequency can also be measured: this is useful for applications that do not have a crystal. The ultra-low-power LSI oscillator has a large manufacturing process deviation. The LSI clock frequency can be measured using the more precise HSI clock source. Using this measurement, a more accurate RTC time base timeouts (when LSI is used as the RTC clock source) and/or an IWDG timeout with an acceptable accuracy can be obtained.

\subsection*{11.4.13 RTC and TAMP clock}

The RTCCLK clock source is used by RTC and TAMP, and can be either the HSE / 32, LSE or LSI clock. It is selected by programming the RTCSEL[1:0] bits in the \textit{RCC Backup domain control register (RCC\_BDCR)}.
control register (RCC_BDCR). This selection cannot be modified without resetting the Backup domain. The system must always be configured so as to get a PCLK frequency greater than or equal to the RTCCLK frequency for a proper operation of the RTC. The TAMP does not require any kernel clock if only the backup registers are used, with tampers in edge detection mode. All other tamper detection modes require a kernel clock (refer to Section 47: Tamper and backup registers (TAMP) for more details).

The LSE is in the Backup domain, whereas the HSE and LSI clocks are not. Consequently:
- If LSE is selected as RTC and TAMP clock, these peripherals continue to work even if the V_{DD} supply is switched off, provided the V_{BAT} supply is maintained.
- If the HSE clock divided by a prescaler is used as the RTC or TAMP clock, the RTC state is not guaranteed if the V_{DD} supply is powered off or if the internal voltage regulator is powered off (removing power from the core domain). Depending on the TAMP configuration, this one can remain functional if used in a mode that does not need any kernel clock.
- If the LSI is used as the RTC or TAMP clock, the RTC state is not guaranteed if the V_{DD} supply is powered off. Depending on the TAMP configuration, this one can remain functional if used in a mode that does not need any kernel clock.

When the RTC and TAMP clock is LSE or LSI, the RTC remains clocked and functional under system reset.

11.4.14 Timer clock

The timer clock frequencies are automatically defined by hardware.

There are two cases:
- If the APB prescaler equals 1, the timer clock frequencies are set to the APB domain frequency.
- Otherwise, they are set to twice (×2) the APB domain frequency.

11.4.15 Watchdog clock

If the independent watchdog (IWDG) is started by either hardware option or software access, the LSI oscillator is forced on and cannot be disabled. After the LSI oscillator temporization, the LSI 32 kHz clock is provided to the IWDG.

11.4.16 Peripherals clock gating and autonomous mode

Peripherals clock gating in Run mode

Each peripheral clock can be enabled by the corresponding EN bit in the RCC_AHBxENR and RCC_APBxENR registers.

When the peripheral clock is not active, read or write accesses to the peripheral registers are not supported.

The enable bit has a synchronization mechanism to create a glitch-free clock for the peripheral. After the enable bit is set, the clock is active after 2 cycles of the peripheral bus clock.

Caution: Just after enabling the clock for a peripheral, the software must wait for these 2 clock cycles before accessing the peripheral registers.
Peripherals clock gating in Sleep mode

When a peripheral is enabled, its clock can be automatically gated off when the device is in Sleep mode, by clearing the peripheral LPEN bit in the RCC_AHBxLPENR and RCC_APBxLPENR registers. Both EN and LPEN bit of the peripheral must be set to keep the clock on in Sleep mode.

11.5 RCC security and privilege functional description

11.5.1 RCC TrustZone security protection modes

When the TrustZone security is activated by the TZEN option byte in the Flash option byte configuration register, the RCC is able to secure RCC configuration and status bits from being modified by non-secure accesses.

This is configured through the RCC secure configuration register (RCC_SECCFGR) to prevent non-secure access to read or modify the following features:

- HSE, HSE-CSS, HSI, CSI, LSI, LSE, LSE-CSS, LSCO, HSI48 configuration and status bits
- PLL1, PLL2, PLL3, AHB and APB prescalers configuration and status bits
- system clock (SYSCLK) and ICLK source clock selection and status bits
- MCO clock output configuration and STOPWUCK and STOPKERWUCK bit
- Remove reset flag RMVF configuration

If SPRIV is set in the RCC privilege configuration register (RCC_PRIVCFGR), the RCC_SECCFGR register can be written only by secure and privileged access. If SPRIV is cleared in RCC_PRIVCFGR, RCC_SECCFGR can be written only by secure access, privileged or unprivileged.

RCC_SECCFGR can be read by secure, non-secure, privileged and unprivileged access.

When a peripheral is configured as secure, its related clock, reset, clock source selection and clock enable during low-power modes control bits, are also secure in the RCC_AHBxENR, RCC_APBxENR, RCC_CCIPRx and RCC_BDCR registers.

A peripheral is secure when:

- For securable peripherals by TZSC (see Section 5.4.3: TrustZone security controller (TZSC)), the SEC security bit corresponding to this peripheral is set in the GTZC TZSC secure configuration registers.
- For TrustZone®-aware peripherals, a security feature of this peripheral is enabled through its dedicated bits.

Table 106 summarizes the RCC secured bits following the security configuration bit in the RCC_SECCFGR register.
When one security configuration bit is set, some configuration and status bits are secured. The RCC registers may contain secure and non-secure bits:

- Secured bits: read and write operations are only allowed by a secure access. Non-secure read returns 0 and write accesses are ignored. No illegal access event is generated.
- Non-secure bits: no restriction. Read and write operations are allowed by both secure and non-secure accesses.
- A non-secure write access to RCC_SECCFGR is ignored and generates an illegal access event. An illegal access interrupt is generated if the RCC illegal access interrupt is enabled in the GTZC TZIC registers. RCC_SECCFGR can be read by secure or non-secure access.

When the TrustZone security is disabled (TZEN = 0xC3), all registers are non-secure. RCC_SECCFGR write accesses are ignored.

Table 106. RCC security configuration summary

<table>
<thead>
<tr>
<th>Configuration bit in RCC_SECCFGR</th>
<th>Secured bits</th>
<th>Corresponding register</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HSISEC</strong></td>
<td>HSION, HSIKERON, HSIRDY</td>
<td>RCC_CR</td>
</tr>
<tr>
<td></td>
<td>HSICAL[11:0], HSITRIM[6:0]</td>
<td>RCC_HSICFGFR</td>
</tr>
<tr>
<td></td>
<td>HSIRDYIE</td>
<td>RCC_CIER</td>
</tr>
<tr>
<td></td>
<td>HSIRDYIF</td>
<td>RCC_CIFR</td>
</tr>
<tr>
<td></td>
<td>HSIRDYC</td>
<td>RCC_CICR</td>
</tr>
<tr>
<td><strong>HSESEC</strong></td>
<td>HSEON, HSERDY, HSEBYP, HSECSSON, HSEEXT</td>
<td>RCC_CR</td>
</tr>
<tr>
<td></td>
<td>HSERDYIE, HSECSSF</td>
<td>RCC_CIER</td>
</tr>
<tr>
<td></td>
<td>HSERDYIF, HSECSSF</td>
<td>RCC_CIFR</td>
</tr>
<tr>
<td></td>
<td>HSERDYC, HSECSSC</td>
<td>RCC_CICR</td>
</tr>
<tr>
<td><strong>CSISEC</strong></td>
<td>CSISON, CSIKERON, CSISRDY</td>
<td>RCC_CR</td>
</tr>
<tr>
<td></td>
<td>CSICAL[7:0], CSITRIM[5:0]</td>
<td>RCC_CICFGFR</td>
</tr>
<tr>
<td></td>
<td>CSISRDYIE</td>
<td>RCC_CIER</td>
</tr>
<tr>
<td></td>
<td>CSISRDYIF</td>
<td>RCC_CIFR</td>
</tr>
<tr>
<td></td>
<td>CSISRDYIC</td>
<td>RCC_CICR</td>
</tr>
<tr>
<td><strong>LSISEC</strong></td>
<td>LSION, LSIRDY, LSIPREDIV, LSCOSEL, LSCOEN</td>
<td>RCC_BDCR</td>
</tr>
<tr>
<td></td>
<td>LSIRDYIE</td>
<td>RCC_CIER</td>
</tr>
<tr>
<td></td>
<td>LSIRDYIF</td>
<td>RCC_CIFR</td>
</tr>
<tr>
<td></td>
<td>LSIRDYC</td>
<td>RCC_CICR</td>
</tr>
<tr>
<td><strong>LSESEC</strong></td>
<td>LSECSSON, LSECSSD, LSEDRV[1:0], LSEBYP, LSERDY, LSEON, LSEEXT, LSCOSEL, LSCOEN</td>
<td>RCC_BDCR</td>
</tr>
<tr>
<td></td>
<td>LSERDYIE</td>
<td>RCC_CIER</td>
</tr>
<tr>
<td></td>
<td>LSERDYF</td>
<td>RCC_CIFR</td>
</tr>
<tr>
<td></td>
<td>LSERDYC</td>
<td>RCC_CICR</td>
</tr>
</tbody>
</table>
### Table 106. RCC security configuration summary (continued)

<table>
<thead>
<tr>
<th>Configuration bit in RCC_SECCFGR</th>
<th>Secured bits</th>
<th>Corresponding register</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSCLKSEC</td>
<td>SW[1:0], SWS[1:0], STOPWUCK, STOPKERWUCK, MCO1SEL[3:0], MCO1PRE[2:0], MCO2SEL[3:0], MCO2PRE[2:0]</td>
<td>RCC_CFRG1</td>
</tr>
<tr>
<td></td>
<td>SYSTICKSEL[1:0]</td>
<td>RCC_CCIPR4</td>
</tr>
<tr>
<td></td>
<td>VOS[1:0]</td>
<td>PWR_VOSR</td>
</tr>
<tr>
<td></td>
<td>TIMPRE</td>
<td>RCC_CFRG1</td>
</tr>
<tr>
<td>PRESCSEC</td>
<td>HPRE[3:0], PPRE1[2:0], PPRE2[2:0], PPRE3[2:0]</td>
<td>RCC_CFRG2</td>
</tr>
<tr>
<td>PLL1SEC</td>
<td>PLL1SRC[1:0], PLL1RGE[1:0], PLL1FRACEN, PLL1M[3:0], PLL1VCOSEL, PLL1PREN, PLL1QEN, PLL1REN</td>
<td>RCC_PLL1CFGR</td>
</tr>
<tr>
<td></td>
<td>PLL1N[8:0], PLL1P[6:0], PLL1Q[6:0], PLL1R[6:0]</td>
<td>RCC_PLL1DIVR</td>
</tr>
<tr>
<td></td>
<td>PLL1FRACN[12:0]</td>
<td>RCC_PLL1FRACR</td>
</tr>
<tr>
<td></td>
<td>PLL1RDY, PLL1ON</td>
<td>RCC_CCR</td>
</tr>
<tr>
<td></td>
<td>PLL1RDIYIE</td>
<td>RCC_CIIER</td>
</tr>
<tr>
<td></td>
<td>PLL1RDYF</td>
<td>RCC_CIFR</td>
</tr>
<tr>
<td></td>
<td>PLL1RDYC</td>
<td>RCC_CIICR</td>
</tr>
<tr>
<td>PLL2SEC</td>
<td>PLL2SRC[1:0], PLL2RGE[1:0], PLL2FRACEN, PLL2M[3:0], PLL2PREN, PLL2QEN, PLL2REN, PLL2VCOSEL</td>
<td>RCC_PLL2CFGR</td>
</tr>
<tr>
<td></td>
<td>PLL2N[8:0], PLL2P[6:0], PLL2Q[6:0], PLL2R[6:0]</td>
<td>RCC_PLL2DIVR</td>
</tr>
<tr>
<td></td>
<td>PLL2FRACN[12:0]</td>
<td>RCC_PLL2FRACR</td>
</tr>
<tr>
<td></td>
<td>PLL2RDY, PLL2ON</td>
<td>RCC_CCR</td>
</tr>
<tr>
<td></td>
<td>PLL2RDIYIE</td>
<td>RCC_CIIER</td>
</tr>
<tr>
<td></td>
<td>PLL2RDYF</td>
<td>RCC_CIFR</td>
</tr>
<tr>
<td></td>
<td>PLL2RDYC</td>
<td>RCC_CIICR</td>
</tr>
<tr>
<td>PLL3SEC</td>
<td>PLL3SRC[1:0], PLL3RGE[1:0], PLL3FRACEN, PLL3M[3:0], PLL3PREN, PLL3QEN, PLL3REN, PLL3VCOSEL</td>
<td>RCC_PLL3CFGR</td>
</tr>
<tr>
<td></td>
<td>PLL3N[8:0], PLL3P[6:0], PLL3Q[6:0], PLL3R[6:0]</td>
<td>RCC_PLL3DIVR</td>
</tr>
<tr>
<td></td>
<td>PLL3FRACN[12:0]</td>
<td>RCC_PLL3FRACR</td>
</tr>
<tr>
<td></td>
<td>PLL3RDY, PLL3ON</td>
<td>RCC_CCR</td>
</tr>
<tr>
<td></td>
<td>PLL3RDIYIE</td>
<td>RCC_CIIER</td>
</tr>
<tr>
<td></td>
<td>PLL3RDYF</td>
<td>RCC_CIFR</td>
</tr>
<tr>
<td></td>
<td>PLL3RDYC</td>
<td>RCC_CIICR</td>
</tr>
<tr>
<td>HSI48SEC(1)</td>
<td>HSI48ON, HSI48RDY</td>
<td>RCC_CCR</td>
</tr>
<tr>
<td></td>
<td>HSI48CAL[9:0]</td>
<td>RCC_CIIIKR</td>
</tr>
<tr>
<td></td>
<td>HSI48RDIY</td>
<td>RCC_CIIER</td>
</tr>
<tr>
<td></td>
<td>HSI48RDYF</td>
<td>RCC_CIFR</td>
</tr>
<tr>
<td></td>
<td>HSI48RDYC</td>
<td>RCC_CIICR</td>
</tr>
</tbody>
</table>
11.5.2 RCC privilege protection modes

By default, after reset, all RCC registers can be read or written with both privileged and unprivileged access except RCC privilege configuration register (RCC_PRIVCFGR) that can be written with privileged access only. RCC_PRIVCFGR can be read by secure and non-secure, privileged and unprivileged access.

The SPRIV bit in RCC_PRIVCFGR can be written with secure privileged access only. This bit configures the privileged access of all RCC secure functions (as defined by RCC secure configuration register (RCC_SECCFGR) or by the GTZC for securable peripherals, or by the peripheral itself in case of TrustZone-aware peripherals).

When the SPRIV bit is set in RCC_PRIVCFGR:
- Writing the RCC secure bits is possible only with privileged access, including RCC_SECCFGR.
- The RCC secure bits can be read only with privileged access except RCC_SECCFGR and RCC_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged RCC bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

The NSPRIV bit in RCC_PRIVCFGR can be written with privileged access only, secure or non-secure. This bit configures the privileged access of all RCC non-secure functions (as defined by RCC_SECCFGR, or by the GTZC for securable peripherals, or by the peripheral itself in case of TrustZone-aware peripherals).

When the NSPRIV bit is set in RCC_PRIVCFGR:
- Writing the RCC non-secure bits is possible only with privileged access.
- The RCC non-secure bits can be read only with privileged access except RCC_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged RCC bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

11.6 RCC low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep mode stops the CPU clock. The memory interface clocks (flash memory, caches and all SRAM interfaces) can be stopped by software during Sleep mode. The AHB to

<table>
<thead>
<tr>
<th>Configuration bit in RCC_SECCFGR</th>
<th>Secured bits</th>
<th>Corresponding register</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPKERSECCFG</td>
<td>CKERPSEL[1:0]</td>
<td>RCC_CCIPR5</td>
</tr>
<tr>
<td>RMVFSSEC</td>
<td>RMVF</td>
<td>RCC_CSR</td>
</tr>
</tbody>
</table>

1. TRIM field of the HSI48 is located in CRS peripheral. Be sure to secure it using CRSSEC bit in GTZC1 TZSC secure configuration register 1.
APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.

- Stop mode stops all the clocks in the core domain and disable the PLLs, HSI, HSI48, CSI and HSE oscillators. However, HSI or CSI can be switched ON to generate a wakeup interrupt. LSI and LSE remain active in Stop mode.
- Standby mode stops all the clocks in the core domain and disable the PLLs, HSI, HSI48, CSI and HSE oscillators.

The CPU DeepSleep mode can be overridden for debugging by setting the DBG_STOP or DBG_STANDBY bit in the DBGMCU_CR register.

When exiting Stop mode, the system clock is either HSI or CSI, depending on the software configuration of STOPWUCK in the *RCC CPU domain clock configuration register 2 (RCC_CFGR2)*. The frequency (range and user trim) of the HSI is the one configured before entering Stop mode.

The other internal oscillator can be automatically woken up in addition to the one used by the system clock, to avoid waiting for the other oscillator wakeup time when the device is back in Run mode. This is done thanks to STOPKERWUCK in RCC_CFGR1.

When leaving the Standby mode, the system clock is HSI (32 MHz). The user trim is lost.

If a flash memory programming operation is ongoing, Stop or Standby mode entry is delayed until the flash memory interface access is finished. If an access to the APB domain is ongoing, Stop or Standby mode entry is delayed until the APB access is finished.
### 11.7 RCC interrupts

*Table 107* summarizes the interrupt sources and the way to control them.

#### Table 107. Interrupt sources and control

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event flag</th>
<th>Description</th>
<th>Enable control bits</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop, Standby modes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RCC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSIRDYF</td>
<td>LSI ready</td>
<td>LSI ready</td>
<td>LSIRDYIE and LSISEC = 0</td>
<td>Set LSIRDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>LSERDYF</td>
<td>LSE ready</td>
<td>LSE ready</td>
<td>LSERDYIE and LSESEC = 0</td>
<td>Set LSERDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>HSIDRYF</td>
<td>HSI ready</td>
<td>HSI ready</td>
<td>HSIDRYIE and HSISEC = 0</td>
<td>Set HSIRDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>HSERDYF</td>
<td>HSE ready</td>
<td>HSE ready</td>
<td>HSERDYIE and HSESEC = 0</td>
<td>Set HSERDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CSISRDFY</td>
<td>CSIS ready</td>
<td>CSIS ready</td>
<td>CSISRDFYE and CSISEC = 0</td>
<td>Set CSISRDFYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>HSI48RDYF</td>
<td>HSI48 ready</td>
<td>HSI48 ready</td>
<td>HSI48RDYIE and HSI48SEC = 0</td>
<td>Set HSI48RDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>PLL1RDYF</td>
<td>PLL1 ready</td>
<td>PLL1 ready</td>
<td>PLL1RDYIE and PLL1SEC = 0</td>
<td>Set PLL1RDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>PLL2RDYF</td>
<td>PLL2 ready</td>
<td>PLL2 ready</td>
<td>PLL2RDYIE and PLL2SEC = 0</td>
<td>Set PLL2RDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>PLL3RDYF</td>
<td>PLL3 ready</td>
<td>PLL3 ready</td>
<td>PLL3RDYIE and PLL3SEC = 0</td>
<td>Set PLL3RDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>RCC_S</strong> (1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSIRDYF</td>
<td>LSI ready</td>
<td>LSI ready</td>
<td>LSIRDYIE and LSISEC = 1</td>
<td>Set LSIRDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>LSERDYF</td>
<td>LSE ready</td>
<td>LSE ready</td>
<td>LSERDYIE and LSESEC = 1</td>
<td>Set LSERDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>HSIDRYF</td>
<td>HSI ready</td>
<td>HSI ready</td>
<td>HSIDRYIE and HSISEC = 1</td>
<td>Set HSIRDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>HSERDYF</td>
<td>HSE ready</td>
<td>HSE ready</td>
<td>HSERDYIE and HSESEC = 1</td>
<td>Set HSERDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CSISRDFY</td>
<td>CSIS ready</td>
<td>CSIS ready</td>
<td>CSISRDFYE and CSISEC = 1</td>
<td>Set CSISRDFYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>HSI48RDYF</td>
<td>HSI48 ready</td>
<td>HSI48 ready</td>
<td>HSI48RDYIE and HSI48SEC = 1</td>
<td>Set HSI48RDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>PLL1RDYF</td>
<td>PLL1 ready</td>
<td>PLL1 ready</td>
<td>PLL1RDYIE and PLL1SEC = 1</td>
<td>Set PLL1RDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>PLL2RDYF</td>
<td>PLL2 ready</td>
<td>PLL2 ready</td>
<td>PLL2RDYIE and PLL2SEC = 1</td>
<td>Set PLL2RDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>PLL3RDYF</td>
<td>PLL3 ready</td>
<td>PLL3 ready</td>
<td>PLL3RDYIE and PLL3SEC = 1</td>
<td>Set PLL3RDYC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
1. The RCC secure interrupt vector is used only when TrustZone is enabled.
2. The LSE CSS failure event (LSECSSD) is connected to TAMP internal tamper 3. to get the interrupt associated to this event, the internal tamper 3 must be enabled, and the internal tamper 3 interrupt must be enabled. The ITAMP3F, ITAMP3E, ITAMP3IE, and CITAMP3F bits are in the TAMP peripheral.
3. It is not possible to mask this interrupt when the security system feature is enabled (HSECSSON = 1).

Table 107. Interrupt sources and control (continued)

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event flag</th>
<th>Description</th>
<th>Enable control bits</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop, Standby modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMP</td>
<td>ITAMP3F(2)</td>
<td>LSE CSS failure</td>
<td>LSECSSON and ITAMP3E(2) and ITAMP3IE(2)</td>
<td>Set CITAMP3F(2) to 1</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NMI</td>
<td>HSECSSF</td>
<td>HSE CSS failure</td>
<td>(3)</td>
<td>Set HSECSSC to 1</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

1. The RCC secure interrupt vector is used only when TrustZone is enabled.
2. The LSE CSS failure event (LSECSSD) is connected to TAMP internal tamper 3. to get the interrupt associated to this event, the internal tamper 3 must be enabled, and the internal tamper 3 interrupt must be enabled. The ITAMP3F, ITAMP3E, ITAMP3IE, and CITAMP3F bits are in the TAMP peripheral.
3. It is not possible to mask this interrupt when the security system feature is enabled (HSECSSON = 1).
11.8 RCC registers

11.8.1 RCC clock control register (RCC.CR)

Address offset: 0x000
Reset value: 0x0000 002B

| Bit 31-30 | Reserved, must be kept at reset value. |
| Bit 29   | PLL3RDY: PLL3 clock ready flag |
|          | Set by hardware to indicate that the PLL3 is locked. |
|          | 0: PLL3 unlocked (default after reset) |
|          | 1: PLL3 locked |

| Bit 28   | PLL3ON: PLL3 enable |
|          | Set and cleared by software to enable PLL3. |
|          | Cleared by hardware when entering Stop or Standby mode. |
|          | 0: PLL3 OFF (default after reset) |
|          | 1: PLL3 ON |

| Bit 27   | PLL2RDY: PLL2 clock ready flag |
|          | Set by hardware to indicate that the PLL is locked. |
|          | 0: PLL2 unlocked |
|          | 1: PLL2 locked |

| Bit 26   | PLL2ON: PLL2 enable |
|          | Set and cleared by software to enable PLL2. |
|          | Cleared by hardware when entering Stop or Standby mode. |
|          | 0: PLL2 OFF (default after reset) |
|          | 1: PLL2 ON |

| Bit 25   | PLL1RDY: PLL1 clock ready flag |
|          | Set by hardware to indicate that the PLL1 is locked. |
|          | 0: PLL1 unlocked (default after reset) |
|          | 1: PLL1 locked |

| Bit 24   | PLL1ON: PLL1 enable |
|          | Set and cleared by software to enable PLL1. |
|          | Cleared by hardware when entering Stop or Standby mode. Note that the hardware prevents writing this bit to 0, if the PLL1 output is used as the system clock. |
|          | 0: PLL1 OFF (default after reset) |
|          | 1: PLL1 ON |

Bits 23:21 Reserved, must be kept at reset value.
Bit 20 **HSEEXT**: external high speed clock type in Bypass mode
Set and reset by software to select the external clock type (analog or digital).
The external clock must be enabled with the HSEON bit to be used by the device. The HSEEXT bit can be written only if the HSE oscillator is disabled.
0: HSE in analog mode (default after reset)
1: HSE in digital mode

Bit 19 **HSECSSON**: HSE clock security system enable
Set by software to enable clock security system on HSE.
This bit is “set only” (disabled by a system reset or when the system enters in Standby mode). When HSECSSON is set, the clock detector is enabled by hardware when the HSE is ready and disabled by hardware if an oscillator failure is detected.
0: CSS on HSE OFF (clock detector OFF) (default after reset)
1: CSS on HSE ON (clock detector ON if the HSE oscillator is stable, OFF if not).

Bit 18 **HSEBYP**: HSE clock bypass
Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit to be used by the device.
The HSEBYP bit can be written only if the HSE oscillator is disabled.
0: HSE oscillator not bypassed (default after reset)
1: HSE oscillator bypassed with an external clock

Bit 17 **HSERDY**: HSE clock ready flag
Set by hardware to indicate that the HSE oscillator is stable.
0: HSE clock is not ready (default after reset)
1: HSE clock is ready

Bit 16 **HSEON**: HSE clock enable
Set and cleared by software.
Cleared by hardware to stop the HSE when entering Stop or Standby mode.
This bit cannot be cleared if the HSE is used directly (via SW mux) as system clock, or if the HSE is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1).
0: HSE is OFF (default after reset)
1: HSE is ON

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

Bit 13 **HSI48RDY**: HSI48 clock ready flag
Set by hardware to indicate that the HSI48 oscillator is stable.
0: HSI48 clock is not ready (default after reset)
1: HSI48 clock is ready

Bit 12 **HSI48ON**: HSI48 clock enable
Set by software and cleared by software or by the hardware when the system enters to Stop or Standby mode.
0: HSI48 is OFF (default after reset)
1: HSI48 is ON

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CSIKERON**: CSI clock enable in Stop mode
Set and reset by software to force the CSI to ON, even in Stop mode, to be quickly available as kernel clock for some peripherals. This bit has no effect on the value of CSION.
0: no effect on CSI (default after reset)
1: CSI is forced to ON even in Stop mode
Bit 9  **CSIRDY**: CSI clock ready flag
Set by hardware to indicate that the CSI oscillator is stable. This bit is activated only if the RC is enabled by CSION (it is not activated if the CSI is enabled by CSIKERON or by a peripheral request).
0: CSI clock is not ready (default after reset)
1: CSI clock is ready

Bit 8  **CSION**: CSI clock enable
Set and reset by software to enable/disable CSI clock for system and/or peripheral.
Set by hardware to force the CSI to ON when the system leaves Stop mode, if STOPWUCK = 1 or STOPKERWUCK = 1.
This bit cannot be cleared if the CSI is used directly (via SW mux) as system clock, or if the CSI is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1).
0: CSI is OFF (default after reset)
1: CSI is ON

Bits 7:6  Reserved, must be kept at reset value.

Bit 5  **HSIDIVF**: HSI divider flag
Set and reset by hardware.
As a write operation to HSIDIV has not an immediate effect on the frequency, this flag indicates the current status of the HSI divider. HSIDIVF goes immediately to 0 when HSIDIV value is changed, and is set back to 1 when the output frequency matches the value programmed into HSIDIV.
0: new division ratio not yet propagated to hsi_ck , hsi_ker_ck (default after reset)
1: hsi_ck , hsi_ker_ck clock frequency reflects the new HSIDIV value (default register value when the clock setting is completed).

Bits 4:3  **HSIDIV[1:0]**: HSI clock divider
Set and reset by software.
These bits allow selecting a division ratio to configure the wanted HSI clock frequency. The HSIDIV cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set to 1). In that case, the new HSIDIV value is ignored.
00: division by 1, hsi_ck, hsi_ker_ck = 64 MHz
01: division by 2, hsi_ck, hsi_ker_ck = 32 MHz (default after reset)
10: division by 4, hsi_ck, hsi_ker_ck = 16 MHz
11: division by 8, hsi_ck, hsi_ker_ck = 8 MHz

Bit 2  **HSIKERON**: HSI clock enable in Stop mode
Set and reset by software to force the HSI to ON, even in Stop mode, to be quickly available as kernel clock for peripherals. This bit has no effect on the value of HSION.
0: no effect on HSI (default after reset)
1: HSI is forced to ON even in Stop mode

Bit 1  **HSIRDY**: HSI clock ready flag
Set by hardware to indicate that the HSI oscillator is stable.
0: HSI clock is not ready (default after reset)
1: HSI clock is ready
### 11.8.2 RCC HSI calibration register (RCC_HSICFGR)

Address offset: 0x010

Reset value: 0x0040 0XXX

Reset value depends on the flash memory option bytes setting.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:23</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
</tbody>
</table>
| 22:16 | HSiTRIM[6:0] | HSi clock trimming  
Set by software to adjust calibration.  
HSITRIM field is added to the engineering option bytes loaded during reset phase (FLASH_HSI_OPT) to form the calibration trimming value.  
HSICAL = HSiTRIM + FLASH_HSI_OPT.  
After a change of HSITRIM it takes one system clock cycle before the new HSITRIM value is updated.  
*Note: The reset value of the field is 0x40.*  
Bits 15:12 | Reserved | Must be kept at reset value. |
| 11:0 | HSiCAL[11:0] | HSi clock calibration  
Set by hardware by option byte loading during system reset nreset. Adjusted by software through trimming bits HSiTRIM.  
This field represents the sum of engineering option byte calibration and HSITRIM bits values. |
### 11.8.3 RCC clock recovery RC register (RCC_CRRCR)

Address offset: 0x014  
Reset value: 0x0000 0XXX  
Reset value depends on the flash option bytes setting

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>HSI48CAL[9:0]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

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

Bits 9:0 **HSI48CAL[9:0]**: Internal RC 48 MHz clock calibration  
Set by hardware by option-byte loading during system reset NRESET. Read-only.

### 11.8.4 RCC CSI calibration register (RCC_CSICFGR)

Address offset: 0x018  
Reset value: 0x0020 0XXX  
Reset value depends on the flash option bytes setting

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>CSITRIM[5:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 21:16 **CSITRIM[5:0]**: CSI clock trimming  
Set by software to adjust calibration.  
CSITRIM field is added to the engineering option bytes loaded during reset phase (FLASH_CSI_OPT) to form the calibration trimming value.  
CSICAL = CSITRIM + FLASH_CSI_OPT.  
**Note**: The reset value of the field is 0x20.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **CSICAL[7:0]**: CSI clock calibration  
Set by hardware by option-byte loading during system reset NRESET. Adjusted by software through trimming bits CSITRIM.  
This field represents the sum of engineering option byte calibration value and CSITRIM bits value.
11.8.5  RCC clock configuration register1 (RCC_CFGR1)

Address offset: 0x01C
Reset value: 0x0000 0000
Access: 0 ≤ wait state ≤ 2; word, half-word and byte access

One or two wait states are inserted only if the access occurs during clock source switch.

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

Bits 31:29  **MCO2SEL[2:0]**: microcontroller clock output 2
- Set and cleared by software. Clock source selection may generate glitches on MCO2.
- It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.
- 000: system clock selected (sys_ck) (default after reset)
- 001: PLL2 oscillator clock selected (pl2_p_ck)
- 010: HSE clock selected (hse_ck)
- 011: PLL1 clock selected (pl1_p_ck)
- 100: CSI clock selected (csi_ck)
- 101: LSI clock selected (lsi_ck)
- others: reserved

Bits 28:25  **MCO2PRE[3:0]**: MCO2 prescaler
- Set and cleared by software to configure the prescaler of the MCO2. Modification of this prescaler may generate glitches on MCO2. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.
- 0000: prescaler disabled (default after reset)
- 0001: division by 1 (bypass)
- 0010: division by 2
- 0011: division by 3
- 0100: division by 4
- ...
- 1111: division by 15

Bits 24:22  **MCO1SEL[2:0]**: Microcontroller clock output 1
- Set and cleared by software. Clock source selection may generate glitches on MCO1.
- It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.
- 000: HSI clock selected (hsi_ck) (default after reset)
- 001: LSE oscillator clock selected (lse_ck)
- 010: HSE clock selected (hse_ck)
- 011: PLL1 clock selected (pl1_p_ck)
- 100: HSI48 clock selected (hsi48_ck)
- others: reserved
**Bits 21:18**  
**MCO1PRE[3:0]:** MCO1 prescaler  
Set and cleared by software to configure the prescaler of the MCO1. Modification of this prescaler may generate glitches on MCO1. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.  
- **0000:** prescaler disabled (default after reset)  
- **0001:** division by 1 (bypass)  
- **0010:** division by 2  
- **0011:** division by 3  
- **0100:** division by 4  
- ...  
- **1111:** division by 15

**Bits 17:16**  
Reserved, must be kept at reset value.

**Bit 15**  
**TIMPRE: timers clocks prescaler selection**  
This bit is set and reset by software to control the clock frequency of all the timers connected to APB1 and APB2 domains.  
- **0:** The timers kernel clock is equal to \( rcc\_hclk1 \) if PPRE1 or PPRE2 corresponds to a division by 1 or 2, else it is equal to \( 2 \times rcc\_pclk1 \) or \( 2 \times rcc\_pclk2 \) (default after reset)  
- **1:** The timers kernel clock is equal to \( 2 \times rcc\_pclk1 \) or \( 2 \times rcc\_pclk2 \) if PPRE1 or PPRE2 corresponds to a division by 1, 2, or 4, else it is equal to \( 4 \times rcc\_pclk1 \) or \( 4 \times rcc\_pclk2 \)

**Bit 14**  
Reserved, must be kept at reset value.

**Bits 13:8**  
**RTCPRE[5:0]:** HSE division factor for RTC clock  
Set and cleared by software to divide the HSE to generate a clock for RTC.  
Caution: The software must set these bits correctly to ensure that the clock supplied to the RTC is lower than 1 MHz. These bits must be configured if needed before selecting the RTC clock source.  
- **000000:** no clock (default after reset)  
- **000001:** no clock  
- **000010:** HSE/2  
- **000011:** HSE/3  
- **000100:** HSE/4  
- ...  
- **111110:** HSE/62  
- **111111:** HSE/63

**Bit 7**  
**STOPKERWUCK:** kernel clock selection after a wakeup from system Stop  
Set and reset by software to select the kernel wakeup clock from system Stop.  
- **0:** HSI selected as wakeup clock from system Stop (default after reset)  
- **1:** CSI selected as wakeup clock from system Stop

**Bit 6**  
**STOPWUCK:** system clock selection after a wakeup from system Stop  
Set and reset by software to select the system wakeup clock from system Stop. The selected clock is also used as emergency clock for the clock security system (CSS) on HSE.  
- **0:** HSI selected as wakeup clock from system Stop (default after reset)  
- **1:** CSI selected as wakeup clock from system Stop  

Caution: STOPWUCK must not be modified when CSS is enabled (by HSECSSON bit) and the system clock is HSE (SWS = 10) or a switch on HSE is requested (SW = 10).

**Bit 5**  
Reserved, must be kept at reset value.
Bits 4:3 **SWS[1:0]**: system clock switch status
- Set and reset by hardware to indicate which clock source is used as system clock. 00: HSI used as system clock (**hsi_ck**) (default after reset).
- 01: CSI used as system clock (**csi_ck**) (default after reset).
- 10: HSE used as system clock (**hse_ck**)
- 11: PLL1 used as system clock (**pll1_p_ck**)
- others: reserved

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SW[1:0]**: system clock and trace clock switch
- Set and reset by software to select system clock and trace clock sources (**sys_ck**).
- Set by hardware to:
  - force the selection of the HSI or CSI (depending on STOPWUCK selection) when leaving a system Stop mode
  - force the selection of the HSI in case of failure of the HSE when used directly or indirectly as system clock
- 00: HSI selected as system clock (**hsi_ck**)
- 01: CSI selected as system clock (**csi_ck**)
- 10: HSE selected as system clock (**hse_ck**)
- 11: PLL1 selected as system clock (**pll1_p_ck** for **sys_ck**)
- others: reserved

### 11.8.6 RCC CPU domain clock configuration register 2 (RCC_CFGR2)

Address offset: 0x020
Reset value: 0x0000 0000

1 or 2 wait states are inserted only if the access occurs during clock source switch.

From 0 to 15 wait states are inserted if the access occurs when the APB or AHB prescalers values update is ongoing.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
<th>Value</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>APB3 DIS</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>APB2 DIS</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>APB1 DIS</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>AHB4 DIS</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>AHB2 DIS</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>AHB1 DIS</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
<th>Value</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>PPRE3[2:0]</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>PPRE2[2:0]</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>PPRE1[2:0]</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>HPRE[3:0]</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 22 **APB3DIS**: APB3 clock disable value. Set and cleared by software
- This bit can be set to further reduce power consumption, when none of the APB3 peripherals are used and when their clocks are disabled in RCC_APB3ENR. When this bit is set, all the APB3 peripherals clocks are off.
- 0: APB3 clock enabled, distributed to peripherals according to their dedicated clock enable control bits
- 1: APB3 clock disabled
Bit 21 APB2DIS: APB2 clock disable value
This bit can be set to further reduce power consumption, when none of the APB2 peripherals are used and when their clocks are disabled in RCC_APB2ENR. When this bit is set, all the APB2 peripherals clocks are off.
0: APB2 clock enabled, distributed to peripherals according to their dedicated clock enable control bits
1: APB2 clock disabled

Bit 20 APB1DIS: APB1 clock disable value
This bit can be set to further reduce power consumption, when none of the APB1 peripherals (except IWDG) are used and when their clocks are disabled in RCC_APB1ENR. When this bit is set, all the APB1 peripherals clocks are off, except for IWDG.
0: APB1 clock enabled, distributed to peripherals according to their dedicated clock enable control bits
1: APB1 clock disabled

Bit 19 AHB4DIS: AHB4 clock disable
This bit can be set to further reduce power consumption, when none of the AHB4 peripherals from RCC_AHB4ENR are used and when their clocks are disabled in RCC_AHB4ENR. When this bit is set, all the AHB4 peripherals clocks from RCC_AHB4ENR are off.
0: AHB4 clock enabled, distributed to peripherals according to their dedicated clock enable control bits
1: AHB4 clock disabled

Bit 18 Reserved, must be kept at reset value.

Bit 17 AHB2DIS: AHB2 clock disable
This bit can be set to further reduce power consumption, when none of the AHB2 peripherals from RCC_AHB2ENR are used and when their clocks are disabled in RCC_AHB2ENR. When this bit is set, all the AHB2 peripherals clocks are off, except for SRAM2 and SRAM3.
0: AHB2 clock enabled, distributed to peripherals according to their dedicated clock enable control bits
1: AHB2 clock disabled

Bit 16 AHB1DIS: AHB1 clock disable
This bit can be set to further reduce power consumption, when none of the AHB1 peripherals from RCC_AHB1ENR are used and when their clocks are disabled in RCC_AHB1ENR. When this bit is set, all the AHB1 peripherals clocks are off, except for FLASH, BKPSRAM, ICACHE, DCACHE1 and SRAM1.
0: AHB1 clock enabled, distributed to peripherals according to their dedicated clock enable control bits
1: AHB1 clock disabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 PPRE3[2:0]: APB low-speed prescaler (APB3)
Set and reset by software to control APB low-speed clocks division factor.
The clocks are divided with the new prescaler factor from 1 to 16 APB cycles after PPRE3 write.
0xx: rcc_pclk3 = rcc_hclk1
100: rcc_pclk3 = rcc_hclk1 / 2
101: rcc_pclk3 = rcc_hclk1 / 4
110: rcc_pclk3 = rcc_hclk1 / 8
111: rcc_pclk3 = rcc_hclk1 / 16
Bit 11  Reserved, must be kept at reset value.

Bits 10:8  **PPRE2[2:0]**: APB high-speed prescaler (APB2)

Set and reset by software to control APB high-speed clocks division factor.

The clocks are divided with the new prescaler factor from 1 to 16 APB cycles after PPRE2 write.

- 0xx: \( rcc\_pclk2 = rcc\_hclk1 \)
- 100: \( rcc\_pclk2 = rcc\_hclk1 / 2 \)
- 101: \( rcc\_pclk2 = rcc\_hclk1 / 4 \)
- 110: \( rcc\_pclk2 = rcc\_hclk1 / 8 \)
- 111: \( rcc\_pclk2 = rcc\_hclk1 / 16 \)

Bit 7  Reserved, must be kept at reset value.

Bits 6:4  **PPRE1[2:0]**: APB low-speed prescaler (APB1)

Set and reset by software to control the division factor of \( rcc\_pclk1 \).

The clock is divided by the new prescaler factor from 1 to 16 cycles of \( rcc\_hclk1 \) after PPRE write.

- 0xx: \( rcc\_pclk1 = rcc\_hclk1 \) (default after reset)
- 100: \( rcc\_pclk1 = rcc\_hclk1 / 2 \)
- 101: \( rcc\_pclk1 = rcc\_hclk1 / 4 \)
- 110: \( rcc\_pclk1 = rcc\_hclk1 / 8 \)
- 111: \( rcc\_pclk1 = rcc\_hclk1 / 16 \)

Bits 3:0  **HPRE[3:0]**: AHB prescaler

Set and reset by software to control the division factor of \( rcc\_hclk1 \). Changing this division ratio has an impact on the frequency of all bus matrix clocks.

- 0xxx: \( rcc\_hclk = sys\_ck \) (default after reset)
- 1000: \( rcc\_hclk = sys\_ck / 2 \)
- 1001: \( rcc\_hclk = sys\_ck / 4 \)
- 1010: \( rcc\_hclk = sys\_ck / 8 \)
- 1011: \( rcc\_hclk = sys\_ck / 16 \)
- 1100: \( rcc\_hclk = sys\_ck / 64 \)
- 1101: \( rcc\_hclk = sys\_ck / 128 \)
- 1110: \( rcc\_hclk = sys\_ck / 256 \)
- 1111: \( rcc\_hclk = sys\_ck / 512 \)

**Caution:** Be careful when using the voltage scaling. Due to the propagation delay of the new division factor, after a prescaler factor change, and before lowering the \( V_{CORE} \) voltage, this register must be read to check that the new prescaler value has been taken into account.

Depending upon the clock source frequency and the voltage range, the software application must program a correct value in HPRE to make sure that the system frequency does not exceed the maximum frequency.
### 11.8.7 RCC PLL clock source selection register (RCC_PLL1CFGR)

Address offset: 0x028  
Reset value: 0x0000 0000  
Access: no wait state, word, half-word and byte access

<table>
<thead>
<tr>
<th>Bit 31:19</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bit 18    | **PLL1REN**: PLL1 DIVR divider output enable  
Set and reset by software to enable the pll1_r_ck output of the PLL1.  
To save power, DIVR1EN and DIVR1 bits must be set to 0 when the pll1_r_ck is not used.  
This bit can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).  
0: pll1_r_ck output disabled (default after reset)  
1: pll1_r_ck output enabled |

| Bit 17    | **PLL1QEN**: PLL1 DIVQ divider output enable  
Set and reset by software to enable the pll1_q_ck output of the PLL1.  
To save power, when the pll1_q_ck output of the PLL1 is not used, the pll1_q_ck must be disabled.  
This bit can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).  
0: pll1_q_ck output disabled (default after reset)  
1: pll1_q_ck output enabled |

| Bit 16    | **PLL1PEN**: PLL1 DIVP divider output enable  
Set and reset by software to enable the pll1_p_ck output of the PLL1.  
This bit can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).  
To save power, when the pll1_p_ck output of the PLL1 is not used, the pll1_p_ck must be disabled.  
0: pll1_p_ck output disabled (default after reset)  
1: pll1_p_ck output enabled |

Bits 15:14  Reserved, must be kept at reset value.
Bits 13:8 **PLL1M[5:0]**: prescaler for PLL1
Set and cleared by software to configure the prescaler of the PLL1.
The hardware does not allow any modification of this prescaler when PLL1 is enabled (PLL1ON = 1 or PLL1RDY = 1).
To save power when PLL1 is not used, the value of PLL1M must be set to 0.
000000: prescaler disabled (default after reset)
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
...
100000: division by 32
...
111111: division by 63

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **PLL1VCOSEL**: PLL1 VCO selection
Set and reset by software to select the proper VCO frequency range used for PLL1. This bit must be written before enabling the PLL1.
0: wide VCO range: 192 to 836 MHz (default after reset)
1: medium VCO range: 150 to 420 MHz

Bit 4 **PLL1FRACEN**: PLL1 fractional latch enable
Set and reset by software to latch the content of FRACN1 into the sigma-delta modulator. To latch the FRACN1 value into the sigma-delta modulator, PLL1FRACEN must be set to 0, then set to 1. The transition 0 to 1 transfers the content of FRACN1 into the modulator.

Bits 3:2 **PLL1RGE[1:0]**: PLL1 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL1. This bit must be written before enabling the PLL1.
00: PLL1 input (ref1_ck) clock range frequency between 1 and 2 MHz (default after reset)
01: PLL1 input (ref1_ck) clock range frequency between 2 and 4 MHz
10: PLL1 input (ref1_ck) clock range frequency between 4 and 8 MHz
11: PLL1 input (ref1_ck) clock range frequency between 8 and 16 MHz

Bits 1:0 **PLL1SRC[1:0]**: PLL1M and PLLs clock source selection
Set and reset by software to select the PLL clock source. These bits can be written only when all PLLs are disabled.
To save power, when no PLL is used, the value of PLL1SRC must be set to '00'.
00: no clock send to PLL1M divider and PLLs (default after reset).
01: HSI selected as PLL clock (hsi_ck)
10: CSI selected as PLL clock (csi_ck)
11: HSE selected as PLL clock (hse_ck)
### 11.8.8 RCC PLL clock source selection register (RCC_PLL2CFGR)

Address offset: 0x02C
Reset value: 0x0000 0000
Access: no wait state, word, half-word and byte access

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:19** Reserved, must be kept at reset value.

**Bit 18** PLL2REN: PLL2 DIVR divider output enable
- Set and reset by software to enable the pll2_r_ck output of the PLL2.
- To save power, DIVR2EN and DIVR2 bits must be set to 0 when the pll2_r_ck is not used.
  - 0: pll2_r_ck output disabled (default after reset)
  - 1: pll2_r_ck output enabled

**Bit 17** PLL2QEN: PLL2 DIVQ divider output enable
- Set and reset by software to enable the pll2_q_ck output of the PLL2.
- To save power, when the pll2_q_ck output of the PLL2 is not used, the pll2_q_ck must be disabled.
  - 0: pll2_q_ck output disabled (default after reset)
  - 1: pll2_q_ck output enabled

**Bit 16** PLL2PEN: PLL2 DIVP divider output enable
- Set and reset by software to enable the pll2_p_ck output of the PLL2.
- To save power, when the pll2_p_ck output of the PLL2 is not used, the pll2_p_ck must be disabled.
  - 0: pll2_p_ck output disabled (default after reset)
  - 1: pll2_p_ck output enabled

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

**Bits 13:8** PLL2M[5:0]: prescaler for PLL2
- Set and cleared by software to configure the prescaler of the PLL2.
- The hardware does not allow any modification of this prescaler when PLL2 is enabled (PLL2ON = 1 or PLL2RDY = 1).
- To save power when PLL2 is not used, the value of PLL2M must be set to 0.
  - 000000: prescaler disabled (default after reset)
  - 000001: division by 1 (bypass)
  - 000010: division by 2
  - 000011: division by 3
  - ...  
  - 100000: division by 32
  - ...  
  - 111111: division by 63

**Bits 7:6** Reserved, must be kept at reset value.
Bit 5 **PLL2VCOSEL**: PLL2 VCO selection
Set and reset by software to select the proper VCO frequency range used for PLL2.
This bit must be written before enabling the PLL2.
0: wide VCO range 192 to 836 MHz (default after reset)
1: medium VCO range 150 to 420 MHz

Bit 4 **PLL2FRACEN**: PLL2 fractional latch enable
Set and reset by software to enable the pll2_p_ck output of the PLL2.
To save power, when the pll2_p_ck output of the PLL2 is not used, the pll2_p_ck must be disabled.
0: pll2_p_ck output disabled (default after reset)
1: pll2_p_ck output enabled

Bits 3:2 **PLL2RGE[1:0]**: PLL2 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL2.
These bits must be written before enabling the PLL2.
00: PLL2 input (ref2_ck) clock range frequency between 1 and 2 MHz (default after reset)
01: PLL2 input (ref2_ck) clock range frequency between 2 and 4 MHz
10: PLL2 input (ref2_ck) clock range frequency between 4 and 8 MHz
11: PLL2 input (ref2_ck) clock range frequency between 8 and 16 MHz

Bits 1:0 **PLL2SRC[1:0]**: PLL2M and PLLs clock source selection
Set and reset by software to select the PLL clock source.
These bits can be written only when all PLLs are disabled.
To save power, when no PLL is used, the value of PLL2SRC must be set to '00'.
00: no clock send to PLL2M divider and PLLs (default after reset)
01: HSI selected as PLL clock (hsi_ck)
10: CSI selected as PLL clock (csi_ck)
11: HSE selected as PLL clock (hse_ck)

### 11.8.9 RCC PLL clock source selection register (RCC_PLL3CFGR)

Address offset: 0x030  
Reset value: 0x0000 0000  
Access: no wait state, word, half-word and byte access

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

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

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **PLL3REN**: PLL3 DIVR divider output enable
Set and reset by software to enable the pll3_r_ck output of the PLL3.
To save power, DIVR2EN and DIVR2 bits must be set to 0 when the pll3_r_ck is not used.
0: pll3_r_ck output disabled (default after reset)
1: pll3_r_ck output enabled
Bit 17 **PLL3QEN**: PLL3 DIVQ divider output enable
Set and reset by software to enable the pll3_q_ck output of the PLL3.
To save power, when the pll3_q_ck output of the PLL3 is not used, the pll3_q_ck must be disabled.
0: pll3_q_ck output disabled (default after reset)
1: pll3_q_ck output enabled

Bit 16 **PLL3PEN**: PLL3 DIVP divider output enable
Set and reset by software to enable the pll3_p_ck output of the PLL3.
To save power, when the pll3_p_ck output of the PLL3 is not used, the pll3_p_ck must be disabled.
0: pll3_p_ck output disabled (default after reset)
1: pll3_p_ck output enabled

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

Bits 13:8 **PLL3M[5:0]**: prescaler for PLL3
Set and cleared by software to configure the prescaler of the PLL3.
The hardware does not allow any modification of this prescaler when PLL3 is enabled (PLL3ON = 1 or PLL3RDY = 1).
To save power when PLL3 is not used, the value of PLL3M must be set to 0.
000000: prescaler disabled (default after reset)
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
...
100000: division by 32
...
111111: division by 63

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **PLL3VCOSEL**: PLL3 VCO selection
Set and reset by software to select the proper VCO frequency range used for PLL3.
This bit must be written before enabling the PLL3.
0: wide VCO range 192 to 836 MHz (default after reset)
1: medium VCO range 150 to 420 MHz

Bit 4 **PLL3FRACEN**: PLL3 fractional latch enable
Set and reset by software to latch the content of FRACN3 into the sigma-delta modulator.
To latch the FRACN3 value into the sigma-delta modulator, PLL3FRACEN must be set to 0, then set to 1. The transition 0 to 1 transfers the content of FRACN3 into the modulator.

Bits 3:2 **PLL3RGE[1:0]**: PLL3 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL3.
This bit must be written before enabling the PLL3.
00: PLL3 input (ref3_ck) clock range frequency between 1 and 2 MHz (default after reset)
01: PLL3 input (ref3_ck) clock range frequency between 2 and 4 MHz
10: PLL3 input (ref3_ck) clock range frequency between 4 and 8 MHz
11: PLL3 input (ref3_ck) clock range frequency between 8 and 16 MHz
Bits 1:0 **PLL3SRC[1:0]**: PLL3M and PLLs clock source selection

Set and reset by software to select the PLL clock source. These bits can be written only when all PLLs are disabled.

To save power, when no PLL is used, the value of PLL3SRC must be set to '00'.

00: no clock send to PLL3M divider and PLLs (default after reset)
01: HSI selected as PLL clock (hsi_ck)
10: CSI selected as PLL clock (csi_ck)
11: HSE selected as PLL clock (hse_ck)

**11.8.10 RCC PLL1 dividers register (RCC_PLL1DIVR)**

Address offset: 0x034
Reset value: 0x0101 0280
Access: no wait state, word, half-word and byte access

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLL1R[6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLL1Q[6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **PLL1R[6:0]**: PLL1 DIVR division factor

Set and reset by software to control the frequency of the pll1_r_ck clock. These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).

0000000: pll1_r_ck = vco1_ck / 1
0000001: pll1_r_ck = vco1_ck / 2 (default after reset)
0000010: pll1_r_ck = vco1_ck / 3
0000011: pll1_r_ck = vco1_ck / 4
...
1111111: pll1_r_ck = vco1_ck / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **PLL1Q[6:0]**: PLL1 DIVQ division factor

Set and reset by software to control the frequency of the pll1_q_ck clock. These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).

0000000: pll1_q_ck = vco1_ck
0000001: pll1_q_ck = vco1_ck / 2 (default after reset)
0000010: pll1_q_ck = vco1_ck / 3
0000011: pll1_q_ck = vco1_ck / 4
...
1111111: pll1_q_ck = vco1_ck / 128
11.8.11 RCC PLL1 fractional divider register (RCC_PLL1FRACR)

Address offset: 0x038
Reset value: 0x0000 0000
Access: no wait state; word and half-word access

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

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

Bits 31:16 Reserved, must be kept at reset value.
Bits 15:3 PLL1FRACN[12:0]: fractional part of the multiplication factor for PLL1 VCO
Set and reset by software to control the fractional part of the multiplication factor of the VCO.
These bits can be written at any time, allowing dynamic fine-tuning of the PLL1 VCO.

**Caution:** The software must set correctly these bits to insure that the VCO output frequency is in the valid frequency range, that is:
  * 128 to 560 MHz if PLL1VCOSEL = 0
  * 150 to 420 MHz if PLL1VCOSEL = 1
VCO output frequency = Fref1_ck x (PLL1N + (PLL1FRACN / 2^{13})), with
  * PLL1N between 8 and 420
  * PLL1FRACN between 0 and 2^{13} - 1
  * The input frequency Fref1_ck must be between 1 and 16 MHz.
To change the PLL1FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:
  * Set the bit PLL1FRACEN to 0
  * Write the new fractional value into PLL1FRACN
  * Set the bit PLL1FRACEN to 1

Bits 2:0 Reserved, must be kept at reset value.

### 11.8.12 RCC PLL1 dividers register (RCC_PLL2DIVR)

Address offset: 0x03C
Reset value: 0x0101 0280
Access: no wait state, word, half-word and byte access

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

**PLL2P[6:0]**
PLL2 DIVR division factor
Set and reset by software to control the frequency of the pll2_r_ck clock.
These bits can be written only when the PLL1 is disabled (PLL2ON = 0 and PLL2RDY = 0).

- 0000000: pll2_r_ck = vco2_ck
- 0000001: pll2_r_ck = vco2_ck / 2 (default after reset)
- 0000010: pll2_r_ck = vco2_ck / 3
- 0000011: pll2_r_ck = vco2_ck / 4...
- 1111111: pll2_r_ck = vco2_ck / 128

Bit 23 Reserved, must be kept at reset value.
11.8.13 RCC PLL2 fractional divider register (RCC_PLL2FRACR)

Address offset: 0x040

Reset value: 0x0000 0000

Access: no wait state; word and half-word access

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

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

<table>
<thead>
<tr>
<th>PLL2FRACN[12:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw  rw  rw  rw</td>
</tr>
<tr>
<td>rw  rw  rw  rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.
Bits 15:3 **PLL2FRACN[12:0]**: fractional part of the multiplication factor for PLL2 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO. These bits can be written at any time, allowing dynamic fine-tuning of the PLL2 VCO.

**Caution:** The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:

* 128 to 560 MHz if PLL2VCOSEL = 0
* 150 to 420 MHz if PLL2VCOSEL = 1

VCO output frequency = Fref2_ck \( \times \) (PLL2N + (PLL2FRACN / 213)), with

* PLL2N between 8 and 420
* PLL2FRACN can be between 0 and 213 - 1
* The input frequency Fref2_ck must be between 1 and 16 MHz.

To change the PLL2FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

* Set the bit PLL2FRACEN to 0
* Write the new fractional value into PLL2FRACN
* Set the bit PLL2FRACEN to 1

Bits 2:0 Reserved, must be kept at reset value.

### 11.8.14 RCC PLL3 dividers register (RCC_PLL3DIVR)

Address offset: 0x044

Reset value: 0x0101 0280

Access: no wait state; word, half-word and byte access

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

Bits 31 Reserved, must be kept at reset value.

Bits 30:24 **PLL3R[6:0]**: PLL3 DIVR division factor

Set and reset by software to control the frequency of the pll3_r_ck clock. These bits can be written only when the PLL1 is disabled (PLL3ON = 0 and PLL3RDY = 0).

0000000: pll3_r_ck = vco3_ck
0000001: pll3_r_ck = vco3_ck / 2 (default after reset)
0000010: pll3_r_ck = vco3_ck / 3
0000011: pll3_r_ck = vco3_ck / 4
...
1111111: pll3_r_ck = vco3_ck / 128

Bit 23 Reserved, must be kept at reset value.
11.8.15 RCC PLL3 fractional divider register (RCC_PLL3FRACR)

Address offset: 0x048

Reset value: 0x0000 0000

Access: no wait state; word and half-word access

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

<table>
<thead>
<tr>
<th>PLL3FRACN[12:0]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
</table>

Bits 31:16 Reserved, must be kept at reset value.
**11.8.16 RCC clock source interrupt enable register (RCC_CIER)**

Address offset: 0x050  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>PLL3FRACN</td>
<td>PLL2FRACN</td>
<td>PLL1FRACN</td>
<td>HSI48FRACN</td>
<td>HSERDFRACN</td>
<td>HSIRDFRACN</td>
<td>CSIRDYIE</td>
<td>LSERDYIE</td>
<td>LSIRDYIE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

| Bits 31:9 | Reserved, must be kept at reset value. |

**Bit 8 PLL3RDYIE**: PLL3 ready interrupt enable  
Set and reset by software to enable/disable interrupt caused by PLL3 lock.  
0: PLL3 lock interrupt disabled (default after reset)  
1: PLL3 lock interrupt enabled

**Bit 7 PLL2RDYIE**: PLL2 ready interrupt enable  
Set and reset by software to enable/disable interrupt caused by PLL2 lock.  
0: PLL2 lock interrupt disabled (default after reset)  
1: PLL2 lock interrupt enabled

**Bit 6 PLL1RDYIE**: PLL1 ready interrupt enable  
Set and reset by software to enable/disable interrupt caused by PLL1 lock.  
0: PLL1 lock interrupt disabled (default after reset)  
1: PLL1 lock interrupt enabled

**Bit 5 HSI48RDYIE**: HSI48 ready interrupt enable  
Set and reset by software to enable/disable interrupt caused by the HSI48 oscillator stabilization.  
0: HSI48 ready interrupt disabled (default after reset)  
1: HSI48 ready interrupt enabled
Bit 4 **HSERDYIE**: HSE ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the HSE oscillator stabilization.
0: HSE ready interrupt disabled (default after reset)
1: HSE ready interrupt enabled

Bit 3 **HSIRDYIE**: HSI ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the HSI oscillator stabilization.
0: HSI ready interrupt disabled (default after reset)
1: HSI ready interrupt enabled

Bit 2 **CSIRDYIE**: CSI ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the CSI oscillator stabilization.
0: CSI ready interrupt disabled (default after reset)
1: CSI ready interrupt enabled

Bit 1 **LSERDYIE**: LSE ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled (default after reset)
1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE**: LSI ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the LSI oscillator stabilization.
0: LSI ready interrupt disabled (default after reset)
1: LSI ready interrupt enabled

### 11.8.17 RCC clock source interrupt flag register (RCC_CIFR)
Address offset: 0x054
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Bits 31:11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
| 30  | Bit 10  | HSECSSF: HSE clock security system interrupt flag
Set by software by writing HSECSSC bit.
Set by hardware in case of HSE clock failure.
0: no clock security interrupt caused by HSE clock failure (default after reset)
1: clock security interrupt caused by HSE clock failure |
| 29  | Bit 9  | Reserved, must be kept at reset value. |
| 28  | Bit 8  | PLL3RDYF: PLL3 ready interrupt flag
Set by software by writing PLL3RDYC bit.
Set by hardware when the PLL3 locks and PLL3RDYIE is set.
0: no clock ready interrupt caused by PLL3 lock (default after reset)
1: clock ready interrupt caused by PLL3 lock |

Payload: 
```
4 10 13 12 11 10 9 8 7 6 5 4 3 2 1 0
r r r r r r r r r r r r
```
Bit 7  PLL2RDYF: PLL2 ready interrupt flag
   Reset by software by writing PLL2RDYC bit.
   Set by hardware when the PLL2 locks and PLL2RDYIE is set.
   0: no clock ready interrupt caused by PLL2 lock (default after reset)
   1: clock ready interrupt caused by PLL2 lock

Bit 6  PLL1RDYF: PLL1 ready interrupt flag
   Reset by software by writing PLL1RDYC bit.
   Set by hardware when the PLL1 locks and PLL1RDYIE is set.
   0: no clock ready interrupt caused by PLL1 lock (default after reset)
   1: clock ready interrupt caused by PLL1 lock

Bit 5  HSI48RDYF: HSI48 ready interrupt flag
   Reset by software by writing HSI48RDYC bit.
   Set by hardware when the HSI48 clock becomes stable and HSI48RDYIE is set.
   0: no clock ready interrupt caused by the HSI48 oscillator (default after reset)
   1: clock ready interrupt caused by the HSI48 oscillator

Bit 4  HSERDYF: HSE ready interrupt flag
   Reset by software by writing HSERDYC bit.
   Set by hardware when the HSE clock becomes stable and HSERDYIE is set.
   0: no clock ready interrupt caused by the HSE (default after reset)
   1: clock ready interrupt caused by the HSE

Bit 3  HSIRDYF: HSI ready interrupt flag
   Reset by software by writing HSIRDYC bit.
   Set by hardware when the HSI clock becomes stable and HSIRDYIE is set.
   0: no clock ready interrupt caused by the HSI (default after reset)
   1: clock ready interrupt caused by the HSI

Bit 2  CSIRDYF: CSI ready interrupt flag
   Reset by software by writing CSIRDYC bit.
   Set by hardware when the CSI clock becomes stable and CSIRDYIE is set.
   0: no clock ready interrupt caused by the CSI (default after reset)
   1: clock ready interrupt caused by the CSI

Bit 1  LSERDYF: LSE ready interrupt flag
   Reset by software by writing LSERDYC bit.
   Set by hardware when the LSE clock becomes stable and LSERDYIE is set.
   0: no clock ready interrupt caused by the LSE (default after reset)
   1: clock ready interrupt caused by the LSE

Bit 0  LSIRDYF: LSI ready interrupt flag
   Reset by software by writing LSIRDYC bit.
   Set by hardware when the LSI clock becomes stable and LSIRDYIE is set.
   0: no clock ready interrupt caused by the LSI (default after reset)
   1: clock ready interrupt caused by the LSI
### 11.8.18 RCC clock source interrupt clear register (RCC_CICR)

Address offset: 0x058  
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>YC</td>
<td>YC</td>
<td>YC</td>
<td>YC</td>
<td>YC</td>
<td>YC</td>
<td>YC</td>
<td>YC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSECSSC**: HSE clock security system interrupt clear  
Set by software to clear HSECSSF.  
Reset by hardware when clear done.  
0: HSECSSF no effect (default after reset)  
1: HSECSSF cleared

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PLL3RDYC**: PLL3 ready interrupt clear  
Set by software to clear PLL3RDYF.  
Reset by hardware when clear done.  
0: PLL3RDYF no effect (default after reset)  
1: PLL3RDYF cleared

Bit 7 **PLL2RDYC**: PLL2 ready interrupt clear  
Set by software to clear PLL2RDYF.  
Reset by hardware when clear done.  
0: PLL2RDYF no effect (default after reset)  
1: PLL2RDYF cleared

Bit 6 **PLL1RDYC**: PLL1 ready interrupt clear  
Set by software to clear PLL1RDYF.  
Reset by hardware when clear done.  
0: PLL1RDYF no effect (default after reset)  
1: PLL1RDYF cleared

Bit 5 **HSI48RDYC**: HSI48 ready interrupt clear  
Set by software to clear HSI48RDYF.  
Reset by hardware when clear done.  
0: HSI48RDYF no effect (default after reset)  
1: HSI48RDYF cleared

Bit 4 **HSERDYC**: HSE ready interrupt clear  
Set by software to clear HSERDYF.  
Reset by hardware when clear done.  
0: HSERDYF no effect (default after reset)  
1: HSERDYF cleared
**RCC AHB1 reset register (RCC_AHB1RSTR)**

Address offset: 0x060  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value 0 Effect</th>
<th>Value 1 Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td><strong>ETHRST</strong>: ETHRST block reset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:20** Reserved, must be kept at reset value.

**Bit 19** **ETHRST**: ETHRST block reset  
Set and reset by software  
0: does not reset ETHRST block (default after reset)  
1: resets the ETHRST block

**Bit 18** Reserved, must be kept at reset value.

**Bit 17** **RAMCFG_RST**: RAMCFG block reset  
Set and reset by software.  
0: does not reset RAMCFG block (default after reset)  
1: resets RAMCFG block

**Bit 16** Reserved, must be kept at reset value.
11.8.20 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x064
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Default</th>
<th>Reset by</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>DCMI_PSSI_RST</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>DAC1_RST</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>ADC_RST</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>GPIOI_RST</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>GPIOH_RST</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>GPIOG_RST</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>GPIOF_RST</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>GPIOE_RST</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>GPIOD_RST</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>GPIOC_RST</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>GPIOB_RST</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>GPIOA_RST</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **SAESRST**: SAES block reset
- Set and reset by software.
- 0: does not reset SAES block (default after reset)
- 1: resets SAES block

Bit 19 **PKARST**: PKA block reset
- Set and reset by software.
- 0: does not reset PKA block (default after reset)
- 1: resets PKA block
Bit 18 **RNGRST**: RNG block reset
   Set and reset by software.
   0: does not reset RNG block (default after reset)
   1: resets RNG block

Bit 17 **HASHRST**: HASH block reset
   Set and reset by software.
   0: does not reset HASH block (default after reset)
   1: resets HASH block

Bit 16 **AESRST**: AES block reset
   Set and reset by software.
   0: does not reset AES block (default after reset)
   1: resets AES block

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DCMI_PSSIRST**: digital camera interface block reset (DCMI or PSSI depending which interface is active)
   Set and reset by software.
   0: does not reset the DCMI/PSSI block (default after reset)
   1: resets the DCMI/PSSI block

Bit 11 **DAC1RST**: DAC block reset
   Set and reset by software.
   0: does not reset DAC block (default after reset)
   1: resets DAC block

Bit 10 **ADCRST**: ADC1 and 2 blocks reset
   Set and reset by software.
   0: does not reset ADC1 and 2 blocks (default after reset)
   1: resets ADC1 and 2 blocks

Bit 9 Reserved, must be kept at reset value.

Bit 8 **GPIOIRST**: GPIOI block reset
   Set and reset by software.
   0: does not reset the GPIOI block (default after reset)
   1: resets the GPIOI block

Bit 7 **GPIOHRST**: GPIOH block reset
   Set and reset by software.
   0: does not reset the GPIOH block (default after reset)
   1: resets the GPIOH block

Bit 6 **GPIOGRST**: GPIOG block reset
   Set and reset by software.
   0: does not reset the GPIOG block (default after reset)
   1: resets the GPIOG block

Bit 5 **GPIOFRST**: GPIOF block reset
   Set and reset by software.
   0: does not reset the GPIOF block (default after reset)
   1: resets the GPIOF block
11.8.21 RCC AHB4 peripheral reset register (RCC_AHB4RSTR)

Address offset: 0x06C  
Reset value: 0x0000 0000

| Bit 31:21 | Reserved, must be kept at reset value. |
| Bit 20  | OCTOSPI1RST: OCTOSPI1 block reset  |
|          | Set and reset by software. |
|          | 0: does not reset OCTOSPI1 block (default after reset) |
|          | 1: resets OCTOSPI1 block |
| Bit 19:17 | Reserved, must be kept at reset value. |
| Bit 16  | FMC RST: FMC block reset  |
|          | Set and reset by software. |
|          | 0: does not reset FMC block (default after reset) |
|          | 1: resets FMC block |
| Bits 15:13 | Reserved, must be kept at reset value. |
Bit 12 **SDMMC2RST**: SDMMC2 and SDMMC2 delay blocks reset
Set and reset by software.
0: does not reset SDMMC2 and SDMMC2 delay blocks (default after reset)
1: resets SDMMC2 and SDMMC2 delay blocks

Bit 11 **SDMMC1RST**: SDMMC1 and SDMMC1 delay blocks reset
Set and reset by software.
0: does not reset SDMMC1 and SDMMC1 delay blocks (default after reset)
1: resets SDMMC1 and SDMMC1 delay blocks

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **OTFDEC1RST**: OTFDEC1 block reset
Set and reset by software.
0: does not reset OTFDEC1 block (default after reset)
1: resets OTFDEC1 block

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

### 11.8.22 RCC APB1 peripheral low reset register (RCC_APB1LRSTR)

Address offset: 0x074
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>UART8RST: UART8 block reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: does not reset the UART8 block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the UART8 block</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>UART7RST: UART7 block reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: does not reset the UART7 block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the UART7 block</td>
</tr>
</tbody>
</table>

Bit 29 Reserved, must be kept at reset value.

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>CECRST: HDMI-CEC block reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: does not reset the HDMI-CEC block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the HDMI-CEC block</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 27</th>
<th>USART11RST: USART11 block reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: does not reset the USART11 block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the USART11 block</td>
</tr>
</tbody>
</table>
Bit 26 **USART10RST**: USART10 block reset
   Set and reset by software.
   0: does not reset the USART10 block (default after reset)
   1: resets the USART10 block

Bit 25 **USART6RST**: USART6 block reset
   Set and reset by software.
   0: does not reset the USART6 block (default after reset)
   1: resets the USART6 block

Bit 24 **CRSRST**: CRS block reset
   Set and reset by software.
   0: does not reset the CRS block (default after reset)
   1: resets the CRS block

Bit 23 **I3C1RST**: I3C1 block reset
   Set and reset by software.
   0: does not reset the I3C1 block (default after reset)
   1: resets the I3C1 block

Bit 22 **I2C2RST**: I2C2 block reset
   Set and reset by software.
   0: does not reset the I2C2 block (default after reset)
   1: resets the I2C2 block

Bit 21 **I2C1RST**: I2C1 block reset
   Set and reset by software.
   0: does not reset the I2C1 block (default after reset)
   1: resets the I2C1 block

Bit 20 **UART5RST**: UART5 block reset
   Set and reset by software.
   0: does not reset the UART5 block (default after reset)
   1: resets the UART5 block

Bit 19 **UART4RST**: UART4 block reset
   Set and reset by software.
   0: does not reset the UART4 block (default after reset)
   1: resets the UART4 block

Bit 18 **USART3RST**: USART3 block reset
   Set and reset by software.
   0: does not reset the USART3 block (default after reset)
   1: resets the USART3 block

Bit 17 **USART2RST**: USART2 block reset
   Set and reset by software.
   0: does not reset the USART2 block (default after reset)
   1: resets the USART2 block

Bit 16 **Reserved, must be kept at reset value.**

Bit 15 **SPI3RST**: SPI3 block reset
   Set and reset by software.
   0: does not reset the SPI3 block (default after reset)
   1: resets the SPI3 block
Bit 14 **SPI2RST**: SPI2 block reset  
Set and reset by software.  
0: does not reset the SPI2 block (default after reset)  
1: resets the SPI2 block  

Bits 13:9 Reserved, must be kept at reset value.  

Bit 8 **TIM14RST**: TIM14 block reset  
Set and reset by software.  
0: does not reset the TIM14 block (default after reset)  
1: resets the TIM14 block  

Bit 7 **TIM13RST**: TIM13 block reset  
Set and reset by software.  
0: does not reset the TIM13 block (default after reset)  
1: resets the TIM13 block  

Bit 6 **TIM12RST**: TIM12 block reset  
Set and reset by software.  
0: does not reset the TIM12 block (default after reset)  
1: resets the TIM12 block  

Bit 5 **TIM7RST**: TIM7 block reset  
Set and reset by software.  
0: does not reset the TIM7 block (default after reset)  
1: resets the TIM7 block  

Bit 4 **TIM6RST**: TIM6 block reset  
Set and reset by software.  
0: does not reset the TIM6 block (default after reset)  
1: resets the TIM6 block  

Bit 3 **TIM5RST**: TIM5 block reset  
Set and reset by software.  
0: does not reset the TIM5 block (default after reset)  
1: resets the TIM5 block  

Bit 2 **TIM4RST**: TIM4 block reset  
Set and reset by software.  
0: does not reset the TIM4 block (default after reset)  
1: resets the TIM4 block  

Bit 1 **TIM3RST**: TIM3 block reset  
Set and reset by software.  
0: does not reset the TIM3 block (default after reset)  
1: resets the TIM3 block  

Bit 0 **TIM2RST**: TIM2 block reset  
Set and reset by software.  
0: does not reset the TIM2 block (default after reset)  
1: resets the TIM2 block
### 11.8.23 RCC APB1 peripheral high reset register (RCC_APB1HRSTR)

Address offset: 0x078  
Reset value: 0x0000 0000

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

Bits 31:24 Reserved, must be kept at reset value.

**Bit 23** UCPD1RST: UCPD1 block reset
- Set and reset by software.
- 0: does not reset the UCPD block (default after reset)
- 1: resets the UCPD block

Bits 22:10 Reserved, must be kept at reset value.

**Bit 9** FDCANRST: FDCAN1 and FDCAN2 blocks reset
- Set and reset by software.
- 0: does not reset the FDCAN1 and FDCAN2 blocks (default after reset)
- 1: resets the FDCAN1 and FDCAN2 blocks

Bits 8:6 Reserved, must be kept at reset value.

**Bit 5** LPTIM2RST: LPTIM2 block reset
- Set and reset by software.
- 0: does not reset the LPTIM2 block (default after reset)
- 1: resets the LPTIM2 block

Bit 4 Reserved, must be kept at reset value.

**Bit 3** DTSRST: DTS block reset
- Set and reset by software.
- 0: does not reset the DTS block (default after reset)
- 1: resets the DTS block

Bit 2 Reserved, must be kept at reset value.

**Bit 1** UART12RST: UART12 block reset
- Set and reset by software.
- 0: does not reset the UART12 block (default after reset)
- 1: resets the UART12 block

Bit 0 UART9RST: UART9 block reset
- Set and reset by software.
- 0: does not reset UART9 block (default after reset)
- 1: resets UART9 block
### 11.8.24 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x07C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:25</th>
<th>Bits 31:25</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 24</td>
<td><strong>USBRST</strong>: USB block reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: does not reset the USB block (default after reset)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: resets the USB block</td>
<td></td>
</tr>
<tr>
<td>Bit 23</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>Bit 22</td>
<td><strong>SAI2RST</strong>: SAI2 block reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: does not reset the SAI2 block (default after reset)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: resets the SAI2 block</td>
<td></td>
</tr>
<tr>
<td>Bit 21</td>
<td><strong>SAI1RST</strong>: SAI1 block reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: does not reset the SAI1 (default after reset)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: resets the SAI1</td>
<td></td>
</tr>
<tr>
<td>Bit 20</td>
<td><strong>SPI6RST</strong>: SPI6 block reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: does not reset the SPI6 block (default after reset)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: resets the SPI6 block</td>
<td></td>
</tr>
<tr>
<td>Bit 19</td>
<td><strong>SPI4RST</strong>: SPI4 block reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: does not reset the SPI4 block (default after reset)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: resets the SPI4 block</td>
<td></td>
</tr>
<tr>
<td>Bit 18</td>
<td><strong>TIM17RST</strong>: TIM17 block reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: does not reset the TIM17 block (default after reset)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: resets the TIM17 block</td>
<td></td>
</tr>
<tr>
<td>Bit 17</td>
<td><strong>TIM16RST</strong>: TIM16 block reset</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: does not reset the TIM16 block (default after reset)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: resets the TIM16 block</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
</table>
Bit 16 **TIM15RST**: TIM15 block reset
Set and reset by software.
0: does not reset the TIM15 block (default after reset)
1: resets the TIM15 block

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1RST**: USART1 block reset
Set and reset by software.
0: does not reset the USART1 block (default after reset)
1: resets the USART1 block

Bit 13 **TIM8RST**: TIM8 block reset
Set and reset by software.
0: does not reset the TIM8 block (default after reset)
1: resets the TIM8 block

Bit 12 **SPI1RST**: SPI1 block reset
Set and reset by software.
0: does not reset the SPI1 block (default after reset)
1: resets the SPI1 block

Bit 11 **TIM1RST**: TIM1 block reset
Set and reset by software.
0: does not reset the TIM1 block (default after reset)
1: resets the TIM1 block

Bits 10:0 Reserved, must be kept at reset value.

### 11.8.25 RCC APB3 peripheral reset register (RCC_APB3RSTR)

Address offset: 0x080
Reset value: 0x0000 0000

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

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VREFRST**: VREFBUF block reset
Set and reset by software.
0: does not reset the VREFBUF block (default after reset)
1: resets the VREFBUF block

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **LPTIM6RST**: LPTIM6 block reset
Set and reset by software.
0: does not reset the LPTIM6 block (default after reset)
1: resets the LPTIM6 block
Bit 14  LPTIM5RST: LPTIM5 block reset
Set and reset by software.
0: does not reset the LPTIM5 block (default after reset)
1: resets the LPTIM5 block

Bit 13  LPTIM4RST: LPTIM4 block reset
Set and reset by software.
0: does not reset the LPTIM4 block (default after reset)
1: resets the LPTIM4 block

Bit 12  LPTIM3RST: LPTIM3 block reset
Set and reset by software.
0: does not reset the LPTIM3 block (default after reset)
1: resets the LPTIM3 block

Bit 11  LPTIM1RST: LPTIM1 block reset
Set and reset by software.
0: does not reset the LPTIM1 block (default after reset)
1: resets the LPTIM1 block

Bits 10:9 Reserved, must be kept at reset value.

Bit 8  I2C4RST: I2C4 block reset
Set and reset by software.
0: does not reset the I2C4 block (default after reset)
1: resets the I2C4 block

Bit 7  I2C3RST: I2C3 block reset
Set and reset by software.
0: does not reset the I2C3 block (default after reset)
1: resets the I2C3 block

Bit 6  LPUART1RST: LPUART1 block reset
Set and reset by software.
0: does not reset the LPUART1 block (default after reset)
1: resets the LPUART1 block

Bit 5  SPI5RST: SPI5 block reset
Set and reset by software.
0: does not reset the SPI5 block (default after reset)
1: resets the SPI5 block

Bits 4:0 Reserved, must be kept at reset value.
### 11.8.26 RCC AHB1 peripherals clock register (RCC_AHB1ENR)

Address offset: 0x088  
Reset value: 0xD000 0100

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>SRAM1EN</td>
<td>SRAM1 clock enable</td>
<td>0 (disabled)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set and reset by software</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>DCACHEEN</td>
<td>DCACHE clock enable</td>
<td>1 (enabled)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set and reset by software</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>BKPRAMEN</td>
<td>BKPRAM clock enable</td>
<td>0 (disabled)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set and reset by software</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>TZSC1EN</td>
<td>TZSC1 clock enable</td>
<td>1 (enabled)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set and reset by software</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>ETHRXEN</td>
<td>ETHERX clock enable</td>
<td>1 (enabled)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set and reset by software</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>ETHTXEN</td>
<td>ETHTX clock enable</td>
<td>1 (enabled)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set and reset by software</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>ETHEN</td>
<td>ETH clock enable</td>
<td>1 (enabled)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set and reset by software</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>ETHEN</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>TZSC1EN</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>ETHERXEN</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>ETHEN</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>ETHTXEN</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>ETHEN</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

**Bit 31** SRAM1EN: SRAM1 clock enable  
Set and reset by software.  
0: SRAM1 clock disabled  
1: SRAM1 clock enabled (default after reset)  

**Bit 30** DCACHEEN: DCACHE clock enable  
Set and reset by software  
0: DCACHE peripheral clock disabled (default after reset)  
1: DCACHE peripheral clock enabled  

**Bit 29** Reserved, must be kept at reset value.  

**Bit 28** BKPRAMEN: BKPRAM clock enable  
Set and reset by software  
0: BKPRAM peripheral clock disabled (default after reset)  
1: BKPRAM peripheral clock enabled  

**Bits 27:25** Reserved, must be kept at reset value.  

**Bit 24** TZSC1EN: TZSC1 clock enable  
Set and reset by software  
0: TZSC1 peripheral clock disabled (default after reset)  
1: TZSC1 peripheral clock enabled  

**Bits 23:22** Reserved, must be kept at reset value.  

**Bit 21** ETHRXEN: ETHERX clock enable  
Set and reset by software  
0: ETHERX clock disabled (default after reset)  
1: ETHERX clock enabled  

**Bit 20** ETHTXEN: ETHTX clock enable  
Set and reset by software  
0: ETHTX clock disabled (default after reset)  
1: ETHTX clock enabled  

**Bit 19** ETHEN: ETH clock enable  
Set and reset by software  
0: ETH peripheral clock disabled (default after reset)  
1: ETH peripheral clock enabled  

**Bit 18** Reserved, must be kept at reset value.
### Bit 17 **RAMCFGEN**: RAMCFG clock enable
- Set and reset by software.
- 0: RAMCFG peripheral clock disabled (default after reset)
- 1: RAMCFG peripheral clock enabled

### Bit 16 Reserved, must be kept at reset value.

### Bit 15 **FMACEN**: FMAC clock enable
- Set and reset by software.
- 0: FMAC peripheral clock disabled (default after reset)
- 1: FMAC peripheral clock enabled

### Bit 14 **CORDICEN**: CORDIC clock enable
- Set and reset by software.
- 0: CORDIC peripheral clock disabled (default after reset)
- 1: CORDIC peripheral clock enabled

### Bit 13 Reserved, must be kept at reset value.

### Bit 12 **CRCEN**: CRC clock enable
- Set and reset by software.
- 0: CRC peripheral clock disabled (default after reset)
- 1: CRC peripheral clock enabled

### Bits 11:9 Reserved, must be kept at reset value.

### Bit 8 **FLITFEN**: Flash interface clock enable
- Set and reset by software.
- 0: FLASH interface clock disabled
- 1: FLASH interface clock enabled (default after reset)

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

### Bit 1 **GPDMA2EN**: GPDMA2 clock enable
- Set and reset by software.
- 0: GPDMA2 peripheral clock disabled (default after reset)
- 1: GPDMA2 peripheral clock enabled

### Bit 0 **GPDMA1EN**: GPDMA1 clock enable
- Set and reset by software.
- 0: GPDMA1 peripheral clock disabled (default after reset)
- 1: GPDMA1 peripheral clock enabled

## 11.8.27 RCC AHB2 peripheral clock register (RCC_AHB2ENR)

**Address offset**: 0x08C

**Reset value**: 0xC000 0000

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

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

506/3119  RM0481 Rev 1
| Bit 31 | **SRAM3EN**: SRAM3 clock enable  
|--------|--------------------------------|
|        | Set and reset by software.  
|        | 0: SRAM3 clock disabled  
|        | 1: SRAM3 clock enabled (default after reset) |

| Bit 30 | **SRAM2EN**: SRAM2 clock enable  
|--------|--------------------------------|
|        | Set and reset by software.  
|        | 0: SRAM2 clock disabled  
|        | 1: SRAM2 clock enabled (default after reset) |

| Bits 29:21 | Reserved, must be kept at reset value. |

| Bit 20 | **SAESEN**: SAES clock enable  
|--------|--------------------------------|
|        | Set and reset by software.  
|        | 0: SAES peripheral clock disabled (default after reset)  
|        | 1: SAES peripheral clock enabled |

| Bit 19 | **PKAEN**: PKA clock enable  
|--------|--------------------------------|
|        | Set and reset by software.  
|        | 0: PKA peripheral clock disabled (default after reset)  
|        | 1: PKA peripheral clock enabled |

| Bit 18 | **RNGEN**: RNG clock enable  
|--------|--------------------------------|
|        | Set and reset by software.  
|        | 0: RNG peripheral clock disabled (default after reset)  
|        | 1: RNG peripheral clock enabled |

| Bit 17 | **HASHEN**: HASH clock enable  
|--------|--------------------------------|
|        | Set and reset by software.  
|        | 0: HASH peripheral clock disabled (default after reset)  
|        | 1: HASH peripheral clock enabled |

| Bit 16 | **AESEN**: AES clock enable  
|--------|--------------------------------|
|        | Set and reset by software.  
|        | 0: AES peripheral clock disabled (default after reset)  
|        | 1: AES peripheral clock enabled |

| Bits 15:13 | Reserved, must be kept at reset value. |

| Bit 12 | **DCMI_PSSIEN**: digital camera interface clock enable (DCMI or PSSI depending which interface is active)  
|--------|--------------------------------|
|        | Set and reset by software.  
|        | 0: DCMI/PSSI peripheral clock disabled (default after reset)  
|        | 1: DCMI/PSSI peripheral clock enabled |

| Bit 11 | **DAC1EN**: DAC clock enable  
|--------|--------------------------------|
|        | Set and reset by software.  
|        | 0: DAC peripheral clock disabled (default after reset)  
|        | 1: DAC peripheral clock enabled |

| Bit 10 | **ADCEN**: ADC1 and 2 peripherals clock enabled  
|--------|--------------------------------|
|        | Set and reset by software.  
|        | 0: ADC1 and 2 peripherals clock disabled (default after reset)  
|        | 1: ADC1 and 2 peripherals clock enabled |

| Bit 9 | Reserved, must be kept at reset value. |
Bit 8  **GPIOIEN**: GPIOI clock enable
Set and reset by software.
0: GPIOI peripheral clock disabled (default after reset)
1: GPIOI peripheral clock enabled

Bit 7  **GPIOHEN**: GPIOH clock enable
Set and reset by software.
0: GPIOH peripheral clock disabled (default after reset)
1: GPIOH peripheral clock enabled

Bit 6  **GPIOGEN**: GPIOG clock enable
Set and reset by software.
0: GPIOG peripheral clock disabled (default after reset)
1: GPIOG peripheral clock enabled

Bit 5  **GPIOFEN**: GPIOF clock enable
Set and reset by software.
0: GPIOF peripheral clock disabled (default after reset)
1: GPIOF peripheral clock enabled

Bit 4  **GPIOEEN**: GPIOE clock enable
Set and reset by software.
0: GPIOE peripheral clock disabled (default after reset)
1: GPIOE peripheral clock enabled

Bit 3  **GPIODEN**: GPIOD clock enable
Set and reset by software.
0: GPIOD peripheral clock disabled (default after reset)
1: GPIOD peripheral clock enabled

Bit 2  **GPIOCEN**: GPIOC clock enable
Set and reset by software.
0: GPIOC peripheral clock disabled (default after reset)
1: GPIOC peripheral clock enabled

Bit 1  **GPIOBEN**: GPIOB clock enable
Set and reset by software.
0: GPIOB peripheral clock disabled (default after reset)
1: GPIOB peripheral clock enabled

Bit 0  **GPIOAEN**: GPIOA clock enable
Set and reset by software.
0: GPIOA peripheral clock disabled (default after reset)
1: GPIOA peripheral clock enabled
**11.8.28 RCC AHB4 peripheral clock register (RCC_AHB4ENR)**

Address offset: 0x094  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:21</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>OCTOSPI1EN</td>
<td>OCTOSPI1 clock enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: OCTOSPI1 peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: OCTOSPI1 peripheral clock enabled</td>
</tr>
<tr>
<td>19:17</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>FMCEN</td>
<td>FMC clock enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: FMC peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: FMC peripheral clock enabled</td>
</tr>
<tr>
<td>15:13</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>12</td>
<td>SDMMC2EN</td>
<td>SDMMC2 and SDMMC2 delay peripheral clock enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: SDMMC2 and SDMMC2 delay peripherals clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: SDMMC2 and SDMMC2 delay peripherals clock enabled</td>
</tr>
<tr>
<td>11</td>
<td>SDMMC1EN</td>
<td>SDMMC1 and SDMMC1 delay peripheral clock enable reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: SDMMC1 and SDMMC1 delay peripherals clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: SDMMC1 and SDMMC1 delay peripherals clock enabled</td>
</tr>
<tr>
<td>10:8</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>7</td>
<td>OTFDEC1EN</td>
<td>OTFDEC1 clock enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: OTFDEC1 peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: OTFDEC1 peripheral clock enabled</td>
</tr>
<tr>
<td>6:0</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
11.8.29  RCC APB1 peripheral clock register (RCC_APB1LENR)

Address offset: 0x09C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>UART8EN: UART8 clock enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: UART8 peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: UART8 peripheral clock enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>UART7EN: UART7 clock enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: UART7 peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: UART7 peripheral clock enabled</td>
</tr>
</tbody>
</table>

| Bit 29 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>CECEN: HDMI-CEC clock enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: HDMI-CEC peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: HDMI-CEC peripheral clock enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 27</th>
<th>USART11EN: USART11 clock enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: USART11 peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: USART11 peripheral clock enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 26</th>
<th>USART10EN: USART10 clock enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: USART10 peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: USART10 peripheral clock enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 25</th>
<th>USART6EN: USART6 clock enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: USART6 peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: USART6 peripheral clock enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 24</th>
<th>CRSEN: CRS clock enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: CRS peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: CRS peripheral clock enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 23</th>
<th>I3C1EN: I3C1 clock enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: I3C1 peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: I3C1 peripheral clock enabled</td>
</tr>
</tbody>
</table>
Bit 22  **I2C2EN**: I2C2 clock enable  
Set and reset by software.  
0: I2C2 peripheral clock disabled (default after reset)  
1: I2C2 peripheral clock enabled

Bit 21  **I2C1EN**: I2C1 clock enable  
Set and reset by software.  
0: I2C1 peripheral clock disabled (default after reset)  
1: I2C1 peripheral clock enabled

Bit 20  **UART5EN**: UART5 clock enable  
Set and reset by software.  
0: UART5 peripheral clock disabled (default after reset)  
1: UART5 peripheral clock enabled

Bit 19  **UART4EN**: UART4 clock enable  
Set and reset by software.  
0: UART4 peripheral clock disabled (default after reset)  
1: UART4 peripheral clock enabled

Bit 18  **USART3EN**: USART3 clock enable  
Set and reset by software.  
0: USART3 peripheral clock disabled (default after reset)  
1: USART3 peripheral clock enabled

Bit 17  **USART2EN**: USART2 clock enable  
Set and reset by software.  
0: USART2 peripheral clock disabled (default after reset)  
1: USART2 peripheral clock enabled

Bit 16  Reserved, must be kept at reset value.

Bit 15  **SPI3EN**: SPI3 clock enable  
Set and reset by software.  
0: SPI3 peripheral clock disabled (default after reset)  
1: SPI3 peripheral clock enabled

Bit 14  **SPI2EN**: SPI2 clock enable  
Set and reset by software.  
0: SPI2 peripheral clock disabled (default after reset)  
1: SPI2 peripheral clock enabled

Bits 13:12  Reserved, must be kept at reset value.

Bit 11  **WWDGEN**: WWDG clock enable  
Set and reset by software.  
0: WWDG peripheral clock disabled (default after reset)  
1: WWDG peripheral clock enabled

Bits 10:9  Reserved, must be kept at reset value.

Bit 8  **TIM14EN**: TIM14 clock enable  
Set and reset by software.  
0: TIM14 peripheral clock disabled (default after reset)  
1: TIM14 peripheral clock enabled
Bit 7 **TIM13EN**: TIM13 clock enable
Set and reset by software.
0: TIM13 peripheral clock disabled (default after reset)
1: TIM13 peripheral clock enabled

Bit 6 **TIM12EN**: TIM12 clock enable
Set and reset by software.
0: TIM12 peripheral clock disabled (default after reset)
1: TIM12 peripheral clock enabled

Bit 5 **TIM7EN**: TIM7 clock enable
Set and reset by software.
0: TIM7 peripheral clock disabled (default after reset)
1: TIM7 peripheral clock enabled

Bit 4 **TIM6EN**: TIM6 clock enable
Set and reset by software.
0: TIM6 peripheral clock disabled (default after reset)
1: TIM6 peripheral clock enabled

Bit 3 **TIM5EN**: TIM5 clock enable
Set and reset by software.
0: TIM5 peripheral clock disabled (default after reset)
1: TIM5 peripheral clock enabled

Bit 2 **TIM4EN**: TIM4 clock enable
Set and reset by software.
0: TIM4 peripheral clock disabled (default after reset)
1: TIM4 peripheral clock enabled

Bit 1 **TIM3EN**: TIM3 clock enable
Set and reset by software.
0: TIM3 peripheral clock disabled (default after reset)
1: TIM3 peripheral clock enabled

Bit 0 **TIM2EN**: TIM2 clock enable
Set and reset by software.
0: TIM2 peripheral clock disabled (default after reset)
1: TIM2 peripheral clock enabled

### 11.8.30 RCC APB1 peripheral clock register (RCC_APB1HENR)

Address offset: 0x0A0

Reset value: 0x0000 0000

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

512/3119 RM0481 Rev 1
Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UCPD1EN**: UCPD1 clock enable
Set and reset by software.
0: UCPD peripheral clock disabled (default after reset)
1: UCPD peripheral clock enabled

Bits 22:10 Reserved, must be kept at reset value.

Bit 9 **FDCANEN**: FDCAN1 and FDCAN2 peripheral clock enable
Set and reset by software.
0: FDCAN1 and FDCAN2 peripheral clock disabled (default after reset)
1: FDCAN1 and FDCAN2 peripheral clock enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2EN**: LPTIM2 clock enable
Set and reset by software.
0: LPTIM2 peripheral clock disabled (default after reset)
1: LPTIM2 peripheral clock enabled

Bit 4 Reserved, must be kept at reset value.

Bit 3 **DTSEN**: DTS clock enable
Set and reset by software.
0: DTS peripheral clock disabled (default after reset)
1: DTS peripheral clock enabled

Bit 2 Reserved, must be kept at reset value.

Bit 1 **UART12EN**: UART12 clock enable
Set and reset by software.
0: UART12 peripheral clock disabled (default after reset)
1: UART12 peripheral clock enabled

Bit 0 **UART9EN**: UART9 clock enable
Set and reset by software.
0: UART9 peripheral clock disabled (default after reset)
1: resets UART9 peripheral clock enabled

### 11.8.31 RCC APB2 peripheral clock register (RCC_APB2ENR)

Address offset: 0x0A4
Reset value: 0x0000 0000

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

Bits 31:25 Reserved, must be kept at reset value.
Bit 24  **USBEN**: USB clock enable
Set and reset by software.
0: USB peripheral clock disabled (default after reset)
1: USB peripheral clock enabled

Bit 23  Reserved, must be kept at reset value.

Bit 22  **SAI2EN**: SAI2 clock enable
Set and cleared by software.
0: SAI2 clock disabled
1: SAI2 clock enabled

Bit 21  **SAI1EN**: SAI1 clock enable
Set and reset by software.
0: SAI1 peripheral clock disabled (default after reset)
1: SAI1 peripheral clock enabled

Bit 20  **SPI6EN**: SPI6 clock enable
Set and reset by software.
0: SPI6 peripheral clock disabled (default after reset)
1: SPI6 peripheral clock enabled

Bit 19  **SPI4EN**: SPI4 clock enable
Set and reset by software.
0: SPI4 peripheral clock disabled (default after reset)
1: SPI4 peripheral clock enabled

Bit 18  **TIM17EN**: TIM17 clock enable
Set and reset by software.
0: TIM17 peripheral clock disabled (default after reset)
1: TIM17 peripheral clock enabled

Bit 17  **TIM16EN**: TIM16 clock enable
Set and reset by software.
0: TIM16 peripheral clock disabled (default after reset)
1: TIM16 peripheral clock enabled

Bit 16  **TIM15EN**: TIM15 clock enable
Set and reset by software.
0: TIM15 peripheral clock disabled (default after reset)
1: TIM15 peripheral clock enabled

Bit 15  Reserved, must be kept at reset value.

Bit 14  **USART1EN**: USART1 clock enable
Set and reset by software.
0: USART1 peripheral clock disabled (default after reset)
1: USART1 peripheral clock enabled

Bit 13  **TIM8EN**: TIM8 clock enable
Set and reset by software.
0: TIM8 peripheral clock disabled (default after reset)
1: TIM8 peripheral clock enabled

Bit 12  **SPI1EN**: SPI1 clock enable
Set and reset by software.
0: SPI1 peripheral clock disabled (default after reset)
1: SPI1 peripheral clock enabled
11.8.32 RCC APB3 peripheral clock register (RCC_APB3ENR)

Address offset: 0x0A8
Reset value: 0x0000 0000

```
 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
 LPTIM6 EN LPTIM5 EN LPTIM4 EN LPTIM3 EN LPTIM1 EN Res. Res. I2C4 EN I2C3 EN LPUART1 EN SPI5 EN Res. Res. Res. SBS
   rw rw rw rw rw rw rw rw rw rw rw rw rw rw
```

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

Bit 21 RTCAPBEN: RTC APB interface clock enable
Set and reset by software.
0: RTC APB interface clock disabled (default after reset)
1: RTC APB interface clock enabled

Bit 20 VREFBUFEN: VREFBUF clock enable
Set and reset by software.
0: VREFBUF peripheral clock disabled (default after reset)
1: VREFBUF peripheral clock enabled

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 LPTIM6EN: LPTIM6 clock enable
Set and reset by software.
0: LPTIM6 peripheral clock disabled (default after reset)
1: LPTIM6 peripheral clock enabled

Bit 14 LPTIM5EN: LPTIM5 clock enable
Set and reset by software.
0: LPTIM5 peripheral clock disabled (default after reset)
1: LPTIM5 peripheral clock enabled

Bit 13 LPTIM4EN: LPTIM4 clock enable
Set and reset by software.
0: LPTIM4 peripheral clock disabled (default after reset)
1: LPTIM4 peripheral clock enabled

Bit 12 LPTIM3EN: LPTIM3 clock enable
Set and reset by software.
0: LPTIM3 peripheral clock disabled (default after reset)
1: LPTIM3 peripheral clock enabled
### RCC_AHB1 sleep clock register (RCC_AHB1LPENR)

Address offset: 0x0B0

Reset value: 0xF13A D103

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

**Bit 31** SRAM1LPEN: SRAM1 clock enable during Sleep mode
- Set and reset by software
- 0: SRAM1 peripheral clock disabled during Sleep mode
- 1: SRAM1 peripheral clock enabled during Sleep mode (default after reset)
Bit 30  **DCACHELPEN**: DCACHE clock enable during Sleep mode  
Set and reset by software  
0: DCACHE peripheral clock disabled during Sleep mode  
1: DCACHE peripheral clock enabled during Sleep mode (default after reset)

Bit 29  **ICACHELPEN**: ICACHE clock enable during Sleep mode  
Set and reset by software  
0: ICACHE peripheral clock disabled during Sleep mode  
1: ICACHE peripheral clock enabled during Sleep mode (default after reset)

Bit 28  **BKPRAML PEN**: BKPRAM clock enable during Sleep mode  
Set and reset by software  
0: BKPRAM peripheral clock disabled during Sleep mode  
1: BKPRAM peripheral clock enabled during Sleep mode (default after reset)

Bits 27:25 Reserved, must be kept at reset value.

Bit 24  **TZSC1LPEN**: TZSC1 clock enable during Sleep mode  
Set and reset by software  
0: TZSC1 peripheral clock disabled during Sleep mode  
1: TZSC1 peripheral clock enabled during Sleep mode (default after reset)

Bits 23:22 Reserved, must be kept at reset value.

Bit 21  **ETHRXLPEN**: ETHRX clock enable during Sleep mode  
Set and reset by software  
0: ETHRX clock disabled during Sleep mode  
1: ETHRX clock enabled during Sleep mode (default after reset)

Bit 20  **ETHTXL PEN**: ETHTX clock enable during Sleep mode  
Set and reset by software  
0: ETHTX clock disabled during Sleep mode  
1: ETHTX clock enabled during Sleep mode (default after reset)

Bit 19  **ETHLPEN**: ETH clock enable during Sleep mode  
Set and reset by software  
0: ETH peripheral clock disabled during Sleep mode  
1: ETH peripheral clock enabled during Sleep mode (default after reset)

Bit 18 Reserved, must be kept at reset value.

Bit 17  **RAMCFGLPEN**: RAMCFG clock enable during Sleep mode  
Set and reset by software.  
0: RAMCFG peripheral clock disabled during Sleep mode  
1: RAMCFG peripheral clock enabled during Sleep mode (default after reset)

Bit 16 Reserved, must be kept at reset value.

Bit 15  **FMACL PEN**: FMAC clock enable during Sleep mode  
Set and reset by software.  
0: FMAC peripheral clock disabled during Sleep mode  
1: FMAC peripheral clock enabled during Sleep mode (default after reset)

Bit 14  **CORDICLPEN**: CORDIC clock enable during Sleep mode  
Set and reset by software.  
0: CORDIC peripheral clock disabled during Sleep mode  
1: CORDIC peripheral clock enabled during Sleep mode (default after reset)

Bit 13 Reserved, must be kept at reset value.
Bit 12 **CRCLPEN**: CRC clock enable during Sleep mode  
Set and reset by software.  
0: CRC peripheral clock disabled during Sleep mode  
1: CRC peripheral clock enabled during Sleep mode (default after reset)

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLITFLPEN**: Flash interface (FLITF) clock enable during Sleep mode  
Set and reset by software.  
0: FLITF peripheral clock disabled during Sleep mode  
1: FLITF peripheral clock enabled during Sleep mode (default after reset)

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

Bit 1 **GPDMA2LPEN**: GPDMA2 clock enable during Sleep mode  
Set and reset by software.  
0: GPDMA2 peripheral clock disabled during Sleep mode  
1: GPDMA2 peripheral clock enabled during Sleep mode (default after reset)

Bit 0 **GPDMA1LPEN**: GPDMA1 clock enable during Sleep mode  
Set and reset by software.  
0: GPDMA1 peripheral clock disabled during Sleep mode  
1: GPDMA1 peripheral clock enabled during Sleep mode (default after reset)

### 11.8.34 RCC AHB2 sleep clock register (RCC_AHB2LPENR)

Address offset: 0x0B4

Reset value: 0xC01F 1DFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM3LPEN</td>
<td>SRAM2LPEN</td>
<td>SAESPEN</td>
<td>PKALPEN</td>
<td>RNGLPEN</td>
<td>HASHLPEN</td>
<td>AESLPEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>DCMILPEN</td>
<td>DACC1LPEN</td>
<td>ADC2LPEN</td>
<td>Res</td>
<td>GPLOILPEN</td>
<td>GPLOHLPEN</td>
<td>GPLOGLPEN</td>
<td>GPLOTLPEN</td>
<td>GPIOFLPEN</td>
<td>GPIOULPEN</td>
<td>GPIOBLPEN</td>
<td>GPIODLPEN</td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **SRAM3LPEN**: SRAM3 clock enable during Sleep mode  
Set and reset by software.  
0: SRAM3 peripheral clock disabled during Sleep mode  
1: SRAM3 peripheral clock enabled during Sleep mode (default after reset)

Bit 30 **SRAM2LPEN**: SRAM2 clock enable during Sleep mode  
Set and reset by software.  
0: SRAM2 peripheral clock disabled during Sleep mode  
1: SRAM2 peripheral clock enabled during Sleep mode (default after reset)

Bits 29:21 Reserved, must be kept at reset value.

Bit 20 **SAESLPEN**: SAES clock enable during Sleep mode  
Set and reset by software.  
0: SAES peripheral clock disabled during Sleep mode  
1: SAES peripheral clock enabled during Sleep mode (default after reset)
Bit 19 **PKALPEN**: PKA clock enable during Sleep mode  
Set and reset by software.  
0: PKA peripheral clock disabled during Sleep mode  
1: PKA peripheral clock enabled during Sleep mode (default after reset)

Bit 18 **RNGLPEN**: RNG clock enable during Sleep mode  
Set and reset by software.  
0: RNG peripheral clock disabled during Sleep mode  
1: RNG peripheral clock enabled during Sleep mode (default after reset)

Bit 17 **HASHLPEN**: HASH clock enable during Sleep mode  
Set and reset by software.  
0: HASH peripheral clock disabled during Sleep mode  
1: HASH peripheral clock enabled during Sleep mode (default after reset)

Bit 16 **AESLPEN**: AES clock enable during Sleep mode  
Set and reset by software.  
0: AES peripheral clock disabled during Sleep mode  
1: AES peripheral clock enabled during Sleep mode (default after reset)

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DCMI_PSSILPEN**: digital camera interface clock enable during Sleep mode (DCMI or PSSI depending which interface is active)  
Set and reset by software.  
0: DCMI/PSSI peripheral clock disabled during Sleep mode  
1: DCMI/PSSI peripheral clock enabled during Sleep mode (default after reset)

Bit 11 **DAC1LPEN**: DAC clock enable during Sleep mode  
Set and reset by software.  
0: DAC peripheral clock disabled during Sleep mode  
1: DAC peripheral clock enabled during Sleep mode (default after reset)

Bit 10 **ADCLPEN**: ADC1 and 2 peripherals clock enable during Sleep mode  
Set and reset by software.  
0: ADC1 and 2 peripherals clock disabled during Sleep mode  
1: ADC1 and 2 peripherals clock enabled during Sleep mode (default after reset)

Bit 9 Reserved, must be kept at reset value.

Bit 8 **GPIOILPEN**: GPIOI clock enable during Sleep mode  
Set and reset by software.  
0: GPIOI peripheral clock disabled during Sleep mode  
1: GPIOI peripheral clock enabled during Sleep mode (default after reset)

Bit 7 **GPIOHLPEN**: GPIOH clock enable during Sleep mode  
Set and reset by software.  
0: GPIOH peripheral clock disabled during Sleep mode  
1: GPIOH peripheral clock enabled during Sleep mode (default after reset)

Bit 6 **GPIOGLPEN**: GPIOG clock enable during Sleep mode  
Set and reset by software.  
0: GPIOG peripheral clock disabled during Sleep mode  
1: GPIOG peripheral clock enabled during Sleep mode (default after reset)
Bit 5 **GPIOFLPEN**: GPIOF clock enable during Sleep mode  
Set and reset by software.  
0: GPIOF peripheral clock disabled during Sleep mode  
1: GPIOF peripheral clock enabled during Sleep mode (default after reset)

Bit 4 **GPIOELPEN**: GPIOE clock enable during Sleep mode  
Set and reset by software.  
0: GPIOE peripheral clock disabled during Sleep mode  
1: GPIOE peripheral clock enabled during Sleep mode (default after reset)

Bit 3 **GPIODLPEN**: GPIOD clock enable during Sleep mode  
Set and reset by software.  
0: GPIOD peripheral clock disabled during Sleep mode  
1: GPIOD peripheral clock enabled during Sleep mode (default after reset)

Bit 2 **GPIOCLPEN**: GPIOC clock enable during Sleep mode  
Set and reset by software.  
0: GPIOC peripheral clock disabled during Sleep mode  
1: GPIOC peripheral clock enabled during Sleep mode (default after reset)

Bit 1 **GPIOBLPEN**: GPIOB clock enable during Sleep mode  
Set and reset by software.  
0: GPIOB peripheral clock disabled during Sleep mode  
1: GPIOB peripheral clock enabled during Sleep mode (default after reset)

Bit 0 **GPIOALPEN**: GPIOA clock enable during Sleep mode  
Set and reset by software.  
0: GPIOA peripheral clock disabled during Sleep mode  
1: GPIOA peripheral clock enabled during Sleep mode (default after reset)

### 11.8.35 RCC AHB4 sleep clock register (RCC_AHB4LPENR)

Address offset: 0x0BC  
Reset value: 0x0011 1880

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **OCTOSPI1LPEN**: OCTOSPI1 clock enable during Sleep mode  
Set and reset by software.  
0: OCTOSPI1 peripheral clock disabled during Sleep mode  
1: OCTOSPI1 peripheral clock enabled during Sleep mode (default after reset)

Bits 19:17 Reserved, must be kept at reset value.
11.8.36 **RCC APB1 sleep clock register (RCC_APB1LLPENR)**

Address offset: 0x0C4

Reset value: 0xDFFE C9FF

<table>
<thead>
<tr>
<th>UART8 LPEN</th>
<th>UART7 LPEN</th>
<th>CECLPEN</th>
<th>USART11LPEN</th>
<th>USART10LPEN</th>
<th>USARTE6LPEN</th>
<th>CRSLPEN</th>
<th>I2C1LPLPEN</th>
<th>I2C2LPLPEN</th>
<th>I2C1LPEN</th>
<th>UART5LPEN</th>
<th>UART4LPEN</th>
<th>USART3LPEN</th>
<th>USART2LPEN</th>
<th>Res.</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SPI3LPEN</th>
<th>SPI2LPEN</th>
<th>Res.</th>
<th>Res.</th>
<th>WWDG LPEN</th>
<th>TIM14LPEN</th>
<th>TIM13LPEN</th>
<th>TIM12LPEN</th>
<th>TIM7LPEN</th>
<th>TIM6LPEN</th>
<th>TIM5LPEN</th>
<th>TIM4LPEN</th>
<th>TIM3LPEN</th>
<th>TIM2LPEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31 **UART8LPEN**: UART8 clock enable during Sleep mode
Set and reset by software.
0: UART8 peripheral clock disabled during Sleep mode
1: UART8 peripheral clock enabled during Sleep mode (default after reset)

Bit 30 **UART7LPEN**: UART7 clock enable during Sleep mode
Set and reset by software.
0: UART7 peripheral clock disabled during Sleep mode
1: UART7 peripheral clock enabled during Sleep mode (default after reset)

Bit 29 Reserved, must be kept at reset value.
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Default After Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td><strong>CECLPEN</strong>: HDMI-CEC clock enable during Sleep mode</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: HDMI-CEC peripheral clock disabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: HDMI-CEC peripheral clock enabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td><strong>USART11LPEN</strong>: USART11 clock enable during Sleep mode</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: USART11 peripheral clock disabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: USART11 peripheral clock enabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td><strong>USART10LPEN</strong>: USART10 clock enable during Sleep mode</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: USART10 peripheral clock disabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: USART10 peripheral clock enabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td><strong>USART6LPEN</strong>: USART6 clock enable during Sleep mode</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: USART6 peripheral clock disabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: USART6 peripheral clock enabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td><strong>CRSLPEN</strong>: CRS clock enable during Sleep mode</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: CRS peripheral clock disabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: CRS peripheral clock enabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td><strong>I3C1LPEN</strong>: I3C1 clock enable during Sleep mode</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: I3C1 peripheral clock disabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: I3C1 peripheral clock enabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td><strong>I2C2LPEN</strong>: I2C2 clock enable during Sleep mode</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: I2C2 peripheral clock disabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: I2C2 peripheral clock enabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td><strong>I2C1LPEN</strong>: I2C1 clock enable during Sleep mode</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: I2C1 peripheral clock disabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: I2C1 peripheral clock enabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td><strong>UART5LPEN</strong>: UART5 clock enable during Sleep mode</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: UART5 peripheral clock disabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: UART5 peripheral clock enabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td><strong>UART4LPEN</strong>: UART4 clock enable during Sleep mode</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: UART4 peripheral clock disabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: UART4 peripheral clock enabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td><strong>USART3LPEN</strong>: USART3 clock enable during Sleep mode</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: USART3 peripheral clock disabled during Sleep mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: USART3 peripheral clock enabled during Sleep mode</td>
<td></td>
</tr>
</tbody>
</table>
Bit 17 **USART2LPEN**: USART2 clock enable during Sleep mode  
Set and reset by software.  
0: USART2 peripheral clock disabled during Sleep mode  
1: USART2 peripheral clock enabled during Sleep mode (default after reset)

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3LPEN**: SPI3 clock enable during Sleep mode  
Set and reset by software.  
0: SPI3 peripheral clock disabled during Sleep mode  
1: SPI3 peripheral clock enabled during Sleep mode (default after reset)

Bit 14 **SPI2LPEN**: SPI2 clock enable during Sleep mode  
Set and reset by software.  
0: SPI2 peripheral clock disabled during Sleep mode  
1: SPI2 peripheral clock enabled during Sleep mode (default after reset)

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGLPEN**: WWDG clock enable during Sleep mode  
Set and reset by software.  
0: WWDG peripheral clock disabled during Sleep mode  
1: WWDG peripheral clock enabled during Sleep mode (default after reset)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **TIM14LPEN**: TIM14 clock enable during Sleep mode  
Set and reset by software.  
0: TIM14 peripheral clock disabled during Sleep mode  
1: TIM14 peripheral clock enabled during Sleep mode (default after reset)

Bit 7 **TIM13LPEN**: TIM13 clock enable during Sleep mode  
Set and reset by software.  
0: TIM13 peripheral clock disabled during Sleep mode  
1: TIM13 peripheral clock enabled during Sleep mode (default after reset)

Bit 6 **TIM12LPEN**: TIM12 clock enable during Sleep mode  
Set and reset by software.  
0: TIM12 peripheral clock disabled during Sleep mode  
1: TIM12 peripheral clock enabled during Sleep mode (default after reset)

Bit 5 **TIM7LPEN**: TIM7 clock enable during Sleep mode  
Set and reset by software.  
0: TIM7 peripheral clock disabled during Sleep mode  
1: TIM7 peripheral clock enabled during Sleep mode (default after reset)

Bit 4 **TIM6LPEN**: TIM6 clock enable during Sleep mode  
Set and reset by software.  
0: TIM6 peripheral clock disabled during Sleep mode  
1: TIM6 peripheral clock enabled during Sleep mode (default after reset)

Bit 3 **TIM5LPEN**: TIM5 clock enable during Sleep mode  
Set and reset by software.  
0: TIM5 peripheral clock disabled during Sleep mode  
1: TIM5 peripheral clock enabled during Sleep mode (default after reset)
Bit 2  **TIM4LPEN**: TIM4 clock enable during Sleep mode  
Set and reset by software.  
0: TIM4 peripheral clock disabled during Sleep mode  
1: TIM4 peripheral clock enabled during Sleep mode (default after reset)

Bit 1  **TIM3LPEN**: TIM3 clock enable during Sleep mode  
Set and reset by software.  
0: TIM3 peripheral clock disabled during Sleep mode  
1: TIM3 peripheral clock enabled during Sleep mode (default after reset)

Bit 0  **TIM2LPEN**: TIM2 clock enable during Sleep mode  
Set and reset by software.  
0: TIM2 peripheral clock disabled during Sleep mode  
1: TIM2 peripheral clock enabled during Sleep mode (default after reset)

### 11.8.37 RCC APB1 sleep clock register (RCC_APB1HLPENR)

Address offset: 0x0C8  
Reset value: 0x4080 022B

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

Bits 31:24  Reserved, must be kept at reset value.

Bit 23  **UCPD1LPEN**: UCPD1 clock enable during Sleep mode  
Set and reset by software.  
0: UCPD peripheral clock disabled during Sleep mode  
1: UCPD peripheral clock enabled during Sleep mode (default after reset)

Bits 22:10  Reserved, must be kept at reset value.

Bit 9  **FDCANLPEN**: FDCAN1 and FDCAN2 peripheral clock enable during Sleep mode  
Set and reset by software.  
0: FDCAN1 and FDCAN2 peripheral clock disabled during Sleep mode  
1: FDCAN1 and FDCAN2 peripheral clock enabled during Sleep mode (default after reset)

Bits 8:6  Reserved, must be kept at reset value.

Bit 5  **LPTIM2LPEN**: LPTIM2 clock enable during Sleep mode  
Set and reset by software.  
0: LPTIM2 peripheral clock disabled during Sleep mode  
1: LPTIM2 peripheral clock enabled during Sleep mode (default after reset)

Bit 4  Reserved, must be kept at reset value.

Bit 3  **DTSLPEN**: DTS clock enable during Sleep mode  
Set and reset by software.  
0: DTS peripheral clock disabled during Sleep mode  
1: DTS peripheral clock enabled during Sleep mode (default after reset)
### RCC APB2 sleep clock register (RCC_APB2LPENR)

Address offset: 0x0CC  
Reset value: 0x017F 7800

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

- **Bit 24** **USBLPEN**: USB clock enable during Sleep mode  
  - Set and reset by software.  
  - 0: USB peripheral clock disabled during Sleep mode  
  - 1: USB peripheral clock enabled during Sleep mode (default after reset)

- **Bit 23** Reserved, must be kept at reset value.

- **Bit 22** **SAI2LPEN**: SAI2 clock enable during Sleep mode  
  - Set and reset by software.  
  - 0: SAI2 peripheral clock disabled during Sleep mode  
  - 1: SAI2 peripheral clock enabled during Sleep mode (default after reset)

- **Bit 21** **SAI1LPEN**: SAI1 clock enable during Sleep mode  
  - Set and reset by software.  
  - 0: SAI1 peripheral clock disabled during Sleep mode  
  - 1: SAI1 peripheral clock enabled during Sleep mode (default after reset)

- **Bit 20** **SPI6LPEN**: SPI6 clock enable during Sleep mode  
  - Set and reset by software.  
  - 0: SPI6 peripheral clock disabled during Sleep mode  
  - 1: SPI6 peripheral clock enabled during Sleep mode (default after reset)

- **Bit 19** **SPI4LPEN**: SPI4 clock enable during Sleep mode  
  - Set and reset by software.  
  - 0: SPI4 peripheral clock disabled during Sleep mode  
  - 1: SPI4 peripheral clock enabled during Sleep mode (default after reset)
Bit 18 TIM17LPEN: TIM17 clock enable during Sleep mode
Set and reset by software.
0: TIM17 peripheral clock disabled during Sleep mode
1: TIM17 peripheral clock enabled during Sleep mode (default after reset)

Bit 17 TIM16LPEN: TIM16 clock enable during Sleep mode
Set and reset by software.
0: TIM16 peripheral clock disabled during Sleep mode
1: TIM16 peripheral clock enabled during Sleep mode (default after reset)

Bit 16 TIM15LPEN: TIM15 clock enable during Sleep mode
Set and reset by software.
0: TIM15 peripheral clock disabled during Sleep mode
1: TIM15 peripheral clock enabled during Sleep mode (default after reset)

Bit 15 Reserved, must be kept at reset value.

Bit 14 USART1LPEN: USART1 clock enable during Sleep mode
Set and reset by software.
0: USART1 peripheral clock disabled during Sleep mode
1: USART1 peripheral clock enabled during Sleep mode (default after reset)

Bit 13 TIM8LPEN: TIM8 clock enable during Sleep mode
Set and reset by software.
0: TIM8 peripheral clock disabled during Sleep mode
1: TIM8 peripheral clock enabled during Sleep mode (default after reset)

Bit 12 SPI1LPEN: SPI1 clock enable during Sleep mode
Set and reset by software.
0: SPI1 peripheral clock disabled during Sleep mode
1: SPI1 peripheral clock enabled during Sleep mode (default after reset)

Bit 11 TIM1LPEN: TIM1 clock enable during Sleep mode
Set and reset by software.
0: TIM1 peripheral clock disabled during Sleep mode
1: TIM1 peripheral clock enabled during Sleep mode (default after reset)

Bits 10:0 Reserved, must be kept at reset value.

11.8.39 RCC APB3 sleep clock register (RCC_APB3LPENR)
Address offset: 0x0D0
Reset value: 0x0030 F9E2

Bits 31:22 Reserved, must be kept at reset value.
Bit 21 **RTCAPBLPEN**: RTC APB interface clock enable during Sleep mode  
Set and reset by software.  
0: RTC APB interface clock disabled during Sleep mode  
1: RTC APB interface clock enabled during Sleep mode (default after reset)

Bit 20 **VREFLPEN**: VREFBUF clock enable during Sleep mode  
Set and reset by software.  
0: VREFBUF peripheral clock disabled during Sleep mode  
1: VREFBUF peripheral clock enabled during Sleep mode (default after reset)

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **LPTIM6LPEN**: LPTIM6 clock enable during Sleep mode  
Set and reset by software.  
0: LPTIM6 peripheral clock disabled during Sleep mode  
1: LPTIM6 peripheral clock enabled during Sleep mode (default after reset)

Bit 14 **LPTIM5LPEN**: LPTIM5 clock enable during Sleep mode  
Set and reset by software.  
0: LPTIM5 peripheral clock disabled during Sleep mode  
1: LPTIM5 peripheral clock enabled during Sleep mode (default after reset)

Bit 13 **LPTIM4LPEN**: LPTIM4 clock enable during Sleep mode  
Set and reset by software.  
0: LPTIM4 peripheral clock disabled during Sleep mode  
1: LPTIM4 peripheral clock enabled during Sleep mode (default after reset)

Bit 12 **LPTIM3LPEN**: LPTIM3 clock enable during Sleep mode  
Set and reset by software.  
0: LPTIM3 peripheral clock disabled during Sleep mode  
1: LPTIM3 peripheral clock enabled during Sleep mode (default after reset)

Bit 11 **LPTIM1LPEN**: LPTIM1 clock enable during Sleep mode  
Set and reset by software.  
0: LPTIM1 peripheral clock disabled during Sleep mode  
1: LPTIM1 peripheral clock enabled during Sleep mode (default after reset)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **I2C4LPEN**: I2C4 clock enable during Sleep mode  
Set and reset by software.  
0: I2C4 peripheral clock disabled during Sleep mode  
1: I2C4 peripheral clock enabled during Sleep mode (default after reset)

Bit 7 **I2C3LPEN**: I2C3 clock enable during Sleep mode  
Set and reset by software.  
0: I2C3 peripheral clock disabled during Sleep mode  
1: I2C3 peripheral clock enabled during Sleep mode (default after reset)

Bit 6 **LPUART1LPEN**: LPUART1 clock enable during Sleep mode  
Set and reset by software.  
0: LPUART1 peripheral clock disabled during Sleep mode  
1: LPUART1 peripheral clock enabled during Sleep mode (default after reset)

Bit 5 **SPI5LPEN**: SPI5 clock enable during Sleep mode  
Set and reset by software.  
0: SPI5 peripheral clock disabled during Sleep mode  
1: SPI5 peripheral clock enabled during Sleep mode (default after reset)
11.8.40 RCC kernel clock configuration register (RCC_CCIPR1)

Address offset: 0x0D8

Reset value: 0x0000 0000

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

Bit 31 TIMICSEL: TIM12, TIM15 and LPTIM2 input capture source selection
Set and reset by software.
0: No internal clock available for timers input capture (default after reset)
1: hsi_ker_ck/1024, hsi_ker_ck/8 and csi_ker_ck/128 selected for timers input capture

Bit 30 Reserved, must be kept at reset value.

Bits 29:27 UART10SEL[2:0]: UART10 kernel clock source selection
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
others: reserved, the kernel clock is disabled

Bits 26:24 UART9SEL[2:0]: UART9 kernel clock source selection
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
others: reserved, the kernel clock is disabled
Bits 23:21  **UART8SEL[2:0]:** UART8 kernel clock source selection
- 000: rcc_pclk1 selected as kernel clock (default after reset)
- 001: pll2_q_ck selected as kernel clock
- 010: pll3_q_ck selected as kernel clock
- 011: hsi_ker_ck selected as kernel clock
- 100: csi_ker_ck selected as kernel clock
- 101: lse_ck selected as kernel clock
-others: reserved, the kernel clock is disabled

Bits 20:18  **UART7SEL[2:0]:** UART7 kernel clock source selection
- 000: rcc_pclk1 selected as kernel clock (default after reset)
- 001: pll2_q_ck selected as kernel clock
- 010: pll3_q_ck selected as kernel clock
- 011: hsi_ker_ck selected as kernel clock
- 100: csi_ker_ck selected as kernel clock
- 101: lse_ck selected as kernel clock
-others: reserved, the kernel clock is disabled

Bits 17:15  **USART6SEL[2:0]:** USART6 kernel clock source selection
- 000: rcc_pclk1 selected as kernel clock (default after reset)
- 001: pll2_q_ck selected as kernel clock
- 010: pll3_q_ck selected as kernel clock
- 011: hsi_ker_ck selected as kernel clock
- 100: csi_ker_ck selected as kernel clock
- 101: lse_ck selected as kernel clock
-others: reserved, the kernel clock is disabled

Bits 14:12  **UART5SEL[2:0]:** UART5 kernel clock source selection
- 000: rcc_pclk1 selected as kernel clock (default after reset)
- 001: pll2_q_ck selected as kernel clock
- 010: pll3_q_ck selected as kernel clock
- 011: hsi_ker_ck selected as kernel clock
- 100: csi_ker_ck selected as kernel clock
- 101: lse_ck selected as kernel clock
-others: reserved, the kernel clock is disabled

Bits 11:9   **UART4SEL[2:0]:** UART4 kernel clock source selection
- 000: rcc_pclk1 selected as kernel clock (default after reset)
- 001: pll2_q_ck selected as kernel clock
- 010: pll3_q_ck selected as kernel clock
- 011: hsi_ker_ck selected as kernel clock
- 100: csi_ker_ck selected as kernel clock
- 101: lse_ck selected as kernel clock
-others: reserved, the kernel clock is disabled

Bits 8:6    **USART3SEL[2:0]:** USART3 kernel clock source selection
- Set and reset by software.
- 000: rcc_pclk1 selected as kernel clock (default after reset)
- 001: pll2_q_ck selected as kernel clock
- 010: pll3_q_ck selected as kernel clock
- 011: hsi_ker_ck selected as kernel clock
- 100: csi_ker_ck selected as kernel clock
- 101: lse_ck selected as kernel clock
-others: reserved, the kernel clock is disabled
11.8.41 RCC kernel clock configuration register (RCC_CCIPR2)

Address offset: 0x0DC

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Possible Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>LPTIM6SEL[2:0]</td>
<td>LPTIM3SEL[2:0]</td>
</tr>
<tr>
<td>30</td>
<td>LPTIM5SEL[2:0]</td>
<td>LPTIM4SEL[2:0]</td>
</tr>
<tr>
<td>29</td>
<td>LPTIM2SEL[2:0]</td>
<td>LPTIM1SEL[2:0]</td>
</tr>
<tr>
<td>28</td>
<td>UART12SEL[2:0]</td>
<td>UART11SEL[2:0]</td>
</tr>
<tr>
<td>27</td>
<td>LPTIMSEL[2:0]</td>
<td>LPTIMSEL[2:0]</td>
</tr>
<tr>
<td>26</td>
<td>UARTSEL[2:0]</td>
<td>UARTSEL[2:0]</td>
</tr>
<tr>
<td>25</td>
<td>UARTSEL[2:0]</td>
<td>UARTSEL[2:0]</td>
</tr>
<tr>
<td>24</td>
<td>UARTSEL[2:0]</td>
<td>UARTSEL[2:0]</td>
</tr>
<tr>
<td>23</td>
<td>UARTSEL[2:0]</td>
<td>UARTSEL[2:0]</td>
</tr>
<tr>
<td>22</td>
<td>UARTSEL[2:0]</td>
<td>UARTSEL[2:0]</td>
</tr>
<tr>
<td>21</td>
<td>UARTSEL[2:0]</td>
<td>UARTSEL[2:0]</td>
</tr>
<tr>
<td>20</td>
<td>UARTSEL[2:0]</td>
<td>UARTSEL[2:0]</td>
</tr>
<tr>
<td>19</td>
<td>UARTSEL[2:0]</td>
<td>UARTSEL[2:0]</td>
</tr>
<tr>
<td>18</td>
<td>UARTSEL[2:0]</td>
<td>UARTSEL[2:0]</td>
</tr>
<tr>
<td>17</td>
<td>UARTSEL[2:0]</td>
<td>UARTSEL[2:0]</td>
</tr>
<tr>
<td>16</td>
<td>UARTSEL[2:0]</td>
<td>UARTSEL[2:0]</td>
</tr>
</tbody>
</table>

Bit 31: Reserved, must be kept at reset value.

Bits 30:28 LPTIM6SEL[2:0]: LPTIM6 kernel clock source selection
- 000: rcc_pclk3 selected as kernel clock (default after reset)
- 001: pll2_p_ck selected as kernel clock
- 010: pll3_r_ck selected as kernel clock
- 011: lsi_ker_ck selected as kernel clock
- 100: per_ck selected as kernel clock
- 101: per_ck selected as kernel clock
- others: reserved, the kernel clock is disabled

Bit 27: Reserved, must be kept at reset value.
Bits 26:24 **LPTIM5SEL[2:0]**: LPTIM5 kernel clock source selection
- 000: rcc_pclk3 selected as kernel clock (default after reset)
- 001: pll2_p_ck selected as kernel clock
- 010: pll3_r_ck selected as kernel clock
- 011: lse_ker_ck selected as kernel clock
- 100: lsi_ker_ck selected as kernel clock
- 101: per_ck selected as kernel clock
- others: reserved, the kernel clock is disabled

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **LPTIM4SEL[2:0]**: LPTIM4 kernel clock source selection
- 000: rcc_pclk3 selected as kernel clock (default after reset)
- 001: pll2_p_ck selected as kernel clock
- 010: pll3_r_ck selected as kernel clock
- 011: lse_ker_ck selected as kernel clock
- 100: lsi_ker_ck selected as kernel clock
- 101: per_ck selected as kernel clock
- others: reserved, the kernel clock is disabled

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **LPTIM3SEL[2:0]**: LPTIM3 kernel clock source selection
- 000: rcc_pclk3 selected as kernel clock (default after reset)
- 001: pll2_p_ck selected as kernel clock
- 010: pll3_r_ck selected as kernel clock
- 011: lse_ker_ck selected as kernel clock
- 100: lsi_ker_ck selected as kernel clock
- 101: per_ck selected as kernel clock
- others: reserved, the kernel clock is disabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **LPTIM2SEL[2:0]**: LPTIM2 kernel clock source selection
- 000: rcc_pclk1 selected as kernel clock (default after reset)
- 001: pll2_p_ck selected as kernel clock
- 010: pll3_r_ck selected as kernel clock
- 011: lse_ker_ck selected as kernel clock
- 100: lsi_ker_ck selected as kernel clock
- 101: per_ck selected as kernel clock
- others: reserved, the kernel clock is disabled

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **LPTIM1SEL[2:0]**: LPTIM1 kernel clock source selection
- 000: rcc_pclk3 selected as kernel clock (default after reset)
- 001: pll2_p_ck selected as kernel clock
- 010: pll3_r_ck selected as kernel clock
- 011: lse_ker_ck selected as kernel clock
- 100: lsi_ker_ck selected as kernel clock
- 101: per_ck selected as kernel clock
- others: reserved, the kernel clock is disabled

Bit 7 Reserved, must be kept at reset value.
11.8.42 RCC kernel clock configuration register (RCC_CCIPR3)

Address offset: 0x0E0
Reset value: 0x0000 0000

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

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

Bits 26:24 LPUART1SEL[2:0]: LPUART1 kernel clock source selection
000: rcc_pclk3 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
others: reserved, the kernel clock is disabled

Bits 23:18 Reserved, must be kept at reset value.

Bits 6:4 UART12SEL[2:0]: UART12 kernel clock source selection
Set and reset by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
others: reserved, the kernel clock is disabled

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 USART11SEL[2:0]: USART11 kernel clock source selection
Set and reset by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
others: reserved, the kernel clock is disabled

Bits 1:0 SPI6SEL[2:1]: SPI6 kernel clock source selection
Set and reset by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
others: reserved, the kernel clock is disabled
Bits 17:15  **SPI6SEL[2:0]**: SPI6 kernel clock source selection
- 000: rcc_pclk2 selected as kernel clock (default after reset)
- 001: pll2_q_ck selected as kernel clock
- 010: pll3_q_ck selected as kernel clock
- 011: hsi_ker_ck selected as kernel clock
- 100: csi_ker_ck selected as kernel clock
- 101: hse_ck selected as kernel clock
- others: reserved, the kernel clock is disabled

Bits 14:12  **SPI5SEL[2:0]**: SPI5 kernel clock source selection
- 000: rcc_pclk3 selected as kernel clock (default after reset)
- 001: pll2_q_ck selected as kernel clock
- 010: pll3_q_ck selected as kernel clock
- 011: hsi_ker_ck selected as kernel clock
- 100: csi_ker_ck selected as kernel clock
- 101: hse_ck selected as kernel clock
- others: reserved, the kernel clock is disabled

Bits 11:9  **SPI4SEL[2:0]**: SPI4 kernel clock source selection
- 000: rcc_pclk3 selected as kernel clock (default after reset)
- 001: pll2_q_ck selected as kernel clock
- 010: pll3_q_ck selected as kernel clock
- 011: hsi_ker_ck selected as kernel clock
- 100: csi_ker_ck selected as kernel clock
- 101: hse_ck selected as kernel clock
- others: reserved, the kernel clock is disabled

Bits 8:6  **SPI3SEL[2:0]**: SPI3 kernel clock source selection
- Set and reset by software.
- 000: pll1_q_ck selected as kernel clock (default after reset)
- 001: pll2_p_ck selected as kernel clock
- 010: pll3_p_ck selected as kernel clock
- 011: AUDIOCLK selected as kernel clock
- 100: per_ck selected as kernel clock
- others: reserved, the kernel clock is disabled

Bits 5:3  **SPI2SEL[2:0]**: SPI2 kernel clock source selection
- Set and reset by software.
- 000: pll1_q_ck selected as kernel clock (default after reset)
- 001: pll2_p_ck selected as kernel clock
- 010: pll3_p_ck selected as kernel clock
- 011: AUDIOCLK selected as kernel clock
- 100: per_ck selected as kernel clock
- others: reserved, the kernel clock is disabled

Bits 2:0  **SPI1SEL[2:0]**: SPI1 kernel clock source selection
- Set and reset by software.
- 000: pll1_q_ck selected as kernel clock (default after reset)
- 001: pll2_p_ck selected as kernel clock
- 010: pll3_p_ck selected as kernel clock
- 011: AUDIOCLK selected as kernel clock
- 100: per_ck selected as kernel clock
- others: reserved, the kernel clock is disabled
11.8.43 RCC kernel clock configuration register (RCC_CCIPR4)

Address offset: 0x0E4
Reset value: 0x0000 0000

| Bits 31:26 | Reserved, must be kept at reset value. |
| Bits 25:24 | **I3C1SEL[1:0]**: I3C1 kernel clock source selection |
|           | 00: rcc_pclk1 selected as kernel clock (default after reset) |
|           | 01: pll3_r_ck selected as kernel clock |
|           | 10: hsi_ker_ck selected as kernel clock |
|           | 11: no clock selected |

| Bits 23:22 | **I2C4SEL[1:0]**: I2C4 kernel clock source selection |
|           | 00: rcc_pclk3 selected as kernel clock (default after reset) |
|           | 01: pll3_r_ck selected as kernel clock |
|           | 10: hsi_ker_ck selected as kernel clock |
|           | 11: csi_ker_ck selected as kernel clock |

| Bits 21:20 | **I2C3SEL[1:0]**: I2C3 kernel clock source selection |
|           | 00: rcc_pclk3 selected as kernel clock (default after reset) |
|           | 01: pll3_r_ck selected as kernel clock |
|           | 10: hsi_ker_ck selected as kernel clock |
|           | 11: csi_ker_ck selected as kernel clock |

| Bits 19:18 | **I2C2SEL[1:0]**: I2C2 kernel clock source selection |
|           | 00: rcc_pclk1 selected as kernel clock (default after reset) |
|           | 01: pll3_r_ck selected as kernel clock |
|           | 10: hsi_ker_ck selected as kernel clock |
|           | 11: csi_ker_ck selected as kernel clock |

| Bits 17:16 | **I2C1SEL[1:0]**: I2C1 kernel clock source selection |
|           | 00: rcc_pclk1 selected as kernel clock (default after reset) |
|           | 01: pll3_r_ck selected as kernel clock |
|           | 10: hsi_ker_ck selected as kernel clock |
|           | 11: csi_ker_ck selected as kernel clock |

| Bits 15:8  | Reserved, must be kept at reset value. |
| Bit 7     | **SDMMC2SEL**: SDMMC2 kernel clock source selection |
|           | 0: pll1_q_ck selected as kernel clock (default after reset) |
|           | 1: pll2_r_ck selected as kernel clock |
| Bit 6     | **SDMMC1SEL**: SDMMC1 kernel clock source selection |
|           | 0: pll1_q_ck selected as kernel clock (default after reset) |
|           | 1: pll2_r_ck selected as kernel clock |
Bits 5:4 **USBSEL[1:0]**: USB kernel clock source selection
- 00: no clock is selected as kernel clock (default after reset)
- 01: pll1_q_ck selected as kernel clock
- 10: pll3_q_ck selected as kernel clock
- 11: hsi48_ker_ck selected as kernel clock

Bits 3:2 **SYSTICKSEL[1:0]**: SYSTICK clock source selection
- 00: rcc_hclk/8 selected as clock source (default after reset)
- 01: lsi_ker_ck[1] selected as clock source
- 10: lse_ck[1] selected as clock source
- 11: reserved, the kernel clock is disabled

*Note:* rcc_hclk frequency must be four times higher than lsi_ker_ck/lse_ck period (LSI/LSE) ≥ 4 * period (HCLK).

Bits 1:0 **OCTOSPI1SEL[1:0]**: OCTOSPI1 kernel clock source selection
- 00: rcc_hclk4 selected as kernel clock (default after reset)
- 01: pll1_q_ck selected as kernel clock
- 10: pll2_r_ck selected as kernel clock
- 11: per_ck selected as kernel clock

### 11.8.44 RCC kernel clock configuration register (RCC_CCIPR5)

Address offset: 0x0E8
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:30 **CKPERSEL[1:0]**: per_ck clock source selection
- 00: hsi_ker_ck selected as kernel clock (default after reset)
- 01: csi_ker_ck selected as kernel clock
- 10: hse_ck selected as kernel clock
- 11: reserved, the per_ck clock is disabled

Bits 29:22 Reserved, must be kept at reset value.

Bits 21:19 **SAI2SEL[2:0]**: SAI2 kernel clock source selection
- 000: pll1_q_ck selected as kernel clock (default after reset)
- 001: pll2_p_ck selected as kernel clock
- 010: pll3_p_ck selected as kernel clock
- 011: AUDIOCLK selected as kernel clock
- 100: per_ck selected as kernel clock
- others: reserved, the kernel clock is disabled
Bits 18:16  **SAI1SEL[2:0]**: SAI1 kernel clock source selection
- 000: pll1_q_ck selected as kernel clock (default after reset)
- 001: pll2_p_ck selected as kernel clock
- 010: pll3_p_ck selected as kernel clock
- 011: AUDIOCLK selected as kernel clock
- 100: per_ck selected as kernel clock
- others: reserved, the kernel clock is disabled

Bits 15:10  Reserved, must be kept at reset value.

Bits 9:8  **FDCANSEL[1:0]**: FDCAN1 and FDCAN2 kernel clock source selection
- 00: hse_ck selected as kernel clock (default after reset)
- 01: pll1_q_ck selected as kernel clock
- 10: pll2_q_ck selected as kernel clock
- 11: reserved, the kernel clock is disabled

Bits 7:6  **CECSEL[1:0]**: HSMI-CEC kernel clock source selection
- 00: lse_ck selected as kernel clock (default after reset)
- 01: lsi_ker_ck selected as kernel clock
- 10: csi_ker_ck/122 selected as kernel clock
- 11: reserved, the kernel clock is disabled

Bits 5:4  **RNGSEL[1:0]**: RNG kernel clock source selection
- 00: hsi48_ker_ck selected as kernel clock (default after reset)
- 01: pll1_q_ck selected as kernel clock
- 10: lse_ck selected as kernel clock
- 11: lsi_ker_ck selected as kernel clock

Bits 3  **DACSEL**: DAC sample and hold clock
- 0: dac_hold_ck selected as kernel clock (default after reset)
- 1: dac_hold_ck selected as kernel clock

Bits 2:0  **ADDCACSEL[2:0]**: ADC and DAC kernel clock source selection
- 000: rcc_hclk selected as kernel clock (default after reset)
- 001: sys_ck selected as kernel clock
- 010: pll2_r_ck selected as kernel clock
- 011: hse_ck selected as kernel clock
- 100: hsi_ker_ck selected as kernel clock
- 101: csi_ker_ck selected as kernel clock
- others: reserved, the kernel clock is disabled

### 11.8.45 RCC Backup domain control register (RCC_BDCR)

**Address offset**: 0x0F0

**Reset value**: 0x0000 0000

Reset by Backup domain reset.

**Access**: 0 ≤ wait state ≤ 3, word, half-word and byte access.

Wait states are inserted in case of successive accesses to this register.

**Note**: After a system reset, the RCC_BDCR register is write-protected (except bits 27:24 and bit 16). To modify the backup domain bits, the DBP bit in the PWR backup domain control register 1 (PWR_BDCR1) must be set to 1: RCC_BDCR bits (except bits 27:24 and bit 16)
are reset only after a Backup domain reset (see Section 11.3.3: Backup domain reset). Any other reset does not have any effect on these bits.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Default State</th>
<th>Function</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-28</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td><strong>LSIRDY</strong>: LSI oscillator ready</td>
<td>rw</td>
<td>Set and cleared by hardware to indicate when the LSI oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after three internal low-speed oscillator clock cycles. This bit is set when the LSI is used by IWDG or RTC, even if LSION = 0.</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td><strong>LSION</strong>: LSI oscillator enable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td><strong>LSCOSEL</strong>: Low-speed clock output selection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td><strong>LSCOEN</strong>: Low-speed clock output (LSCO) enable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td><strong>VSWRST</strong>: VSwitch domain software reset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td><strong>RTCEN</strong>: RTC clock enable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14-10</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11-8</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9-6</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-2</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-0</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 9:8 **RTCSEL[1:0]**: RTC clock source selection
- Set by software to select the clock source for the RTC.
- These bits can be written only one time (except in case of failure detection on LSE).
- These bits must be written before LSECSSON is enabled.
- The VSWRST bit can be used to reset them, then it can be written one time again.
- If HSE is selected as RTC clock, this clock is lost when the system is in Stop mode or in case of a pin reset (NRST).
- 00: no clock (default after Backup domain reset)
- 01: LSE selected as RTC clock
- 10: LSI selected as RTC clock
- 11: HSE divided by RTCPRE value selected as RTC clock

Bit 7 **LSEEXT**: low-speed external clock type in bypass mode
- Set and reset by software to select the external clock type (analog or digital).
- The external clock must be enabled with the LSEON bit, to be used by the device.
- The LSEEXT bit can be written only if the LSE oscillator is disabled.
- 0: LSE in analog mode (default after Backup domain reset)
- 1: LSE in digital mode (do not use if RTC is active)

Bit 6 **LSECSSD**: LSE clock security system failure detection
- Set by hardware to indicate when a failure has been detected by the clock security system on the external 32 kHz oscillator.
- 0: no failure detected on 32 kHz oscillator (default after Backup domain reset)
- 1: failure detected on 32 kHz oscillator

Bit 5 **LSECSSON**: LSE clock security system enable
- Set by software to enable the clock security system on 32 kHz oscillator.
- LSECSSON must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware) and after RTCSEL is selected.
- Once enabled, this bit cannot be disabled, except after a LSE failure detection (LSECSSD = 1). In that case the software must disable LSECSSON.
- 0: CSS on 32 kHz oscillator OFF (default after Backup domain reset)
- 1: CSS on 32 kHz oscillator ON

Bits 4:3 **LSEDRV[1:0]**: LSE oscillator driving capability
- Set by software to select the driving capability of the LSE oscillator.
- These bit can be written only if LSE oscillator is disabled (LSEON = 0 and LSERDY = 0).
- 00: lowest drive (default after Backup domain reset)
- 01: medium-low drive
- 10: medium-high drive
- 11: highest drive

Bit 2 **LSEBYP**: LSE oscillator bypass
- Set and reset by software to bypass oscillator in debug mode. This bit must not be written when the LSE is enabled (by LSEON or ready (LSERDY = 1)
- 0: LSE oscillator not bypassed (default after Backup domain reset)
- 1: LSE oscillator bypassed

Bit 1 **LSERDY**: LSE oscillator ready
- Set and reset by hardware to indicate when the LSE is stable.
- This bit needs 6 cycles of lse_ck clock to fall down after LSEON has been set to 0.
- 0: LSE oscillator not ready (default after Backup domain reset)
- 1: LSE oscillator ready
Bit 0  **LSEON**: LSE oscillator enabled
- Set and reset by software.
- 0: LSE oscillator OFF (default after Backup domain reset)
- 1: LSE oscillator ON

**11.8.46 RCC reset status register (RCC_RSR)**

Address offset: 0x0F4

Reset value: 0x0C00 0000

Reset by power-on reset only.

Access: $0 \leq$ wait state $\leq 3$, word, half-word and byte access.

Wait states are inserted in case of successive accesses to this register.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>LPWRRSTF: Low-power reset flag</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>WWDRSTF: Window watchdog reset flag</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>IWDGRSTF: Independent watchdog reset flag</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>SFTRSTF: System reset from CPU reset flag</td>
<td>rw</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31  **LPWRRSTF**: Low-power reset flag
- Set by hardware when a reset occurs due to Stop or Standby mode entry, whereas the corresponding nRST_STOP, nRST_STBY option bit is cleared.
- Cleared by writing to the RMVF bit.
- 0: No illegal low-power mode reset occurred
- 1: Illegal low-power mode reset occurred

Bit 30  **WWDRSTF**: window watchdog reset flag
- Reset by software by writing the RMVF bit.
- Set by hardware when a window watchdog reset occurs.
- 0: no window watchdog reset occurred from WWDG (default after power-on reset)
- 1: window watchdog reset occurred from WWDG

Bit 29  **IWDGRSTF**: independent watchdog reset flag
- Reset by software by writing the RMVF bit.
- Set by hardware when an independent watchdog reset occurs.
- 0: no independent watchdog reset occurred (default after power-on reset)
- 1: independent watchdog reset occurred

Bit 28  **SFTRSTF**: system reset from CPU reset flag
- Reset by software by writing the RMVF bit.
- Set by hardware when the system reset is due to CPU. The CPU can generate a system reset by writing SYSRESETREQ bit of AIRCR register of the core M33.
- 0: no CPU software reset occurred (default after power-on reset)
- 1: a system reset has been generated by the CPU
11.8.47 RCC secure configuration register (RCC_SECCFGR)

Address offset: 0x110

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

When the system is secure (TZEN = 0xB4), this register can be written only by a secure privileged access if SPRIV = 1, and by a secure privileged or unprivileged access if SPRIV = 0. A non-secure write access generates an illegal access event and data is not written. This register can be read by secure or non-secure, privilege or unprivileged access. When the system is not secure (TZEN = 0xC3), this register is read as 0 and the register write is ignored.

### Bit 27 BORRSTF: BOR reset flag
- Reset by software by writing the RMVF bit.
- Set by hardware when a BOR reset occurs (pwr_bor_rst).
  - 0: no BOR reset occurred
  - 1: BOR reset occurred (default after power-on reset)

### Bit 26 PINRSTF: pin reset flag (NRST)
- Reset by software by writing the RMVF bit.
- Set by hardware when a reset from pin occurs.
  - 0: no reset from pin occurred
  - 1: reset from pin occurred (default after power-on reset)

Bits 25:24 Reserved, must be kept at reset value.

### Bit 23 RMVF: remove reset flag
- Set and reset by software to reset the value of the reset flags.
  - 0: reset of the reset flags not activated (default after power-on reset)
  - 1: resets the value of the reset flags

Bits 22:0 Reserved, must be kept at reset value.
Bit 11 **HSI48SEC**: HSI48 clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 10 Reserved, must be kept at reset value.

Bit 9 **PLL3SEC**: PLL3 clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 8 **PLL2SEC**: PLL2 clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 7 **PLL1SEC**: PLL1 clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 6 **PRESCSEC**: AHBx/APBx prescaler configuration bits security
Set and reset by software.
0: non secure
1: secure

Bit 5 **SYSCLKSEC**: SYCLK clock selection, STOPWUCK bit, clock output on MCO configuration security
Set and reset by software.
0: non secure
1: secure

Bit 4 **LSESEC**: LSE clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 3 **LSISEC**: LSI clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 2 **CSISEC**: CSI clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 1 **HSESEC**: HSE clock configuration bits, status bits and HSE_CSS security
Set and reset by software.
0: non secure
1: secure

Bit 0 **HSISEC**: HSI clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure
11.8.48  RCC privilege configuration register (RCC_PRIVCFGR)

Address offset: 0x114
Reset value: 0x0000 0000
Access: no wait state; word, half-word and byte access

This register can be written only by a privileged access. It can be read by privileged or unprivileged access.

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

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

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

Bit 1  NSPRIV: RCC non-secure functions privilege configuration

Set and reset by software. This bit can be written only by privileged access, secure or non-secure.
0: Read and write to RCC non-secure functions can be done by privileged or unprivileged access.
1: Read and write to RCC non-secure functions can be done by privileged access only

Bit 0  SPRIV: RCC secure functions privilege configuration

Set and reset by software. This bit can be written only by a secure privileged access.
0: Read and write to RCC secure functions can be done by privileged or unprivileged access.
1: Read and write to RCC secure functions can be done by privileged access only
### 11.9 RCC register map

**Table 108. RCC register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>RCC_CR</td>
<td>00000000000000</td>
<td>0x10</td>
<td>RCC_HSICFGR</td>
<td>10000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td>10000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00000000</td>
<td></td>
<td></td>
<td>10000000000000</td>
</tr>
<tr>
<td>0x010</td>
<td>RCC_CRRCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>RCC_CRRCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x018</td>
<td>RCC_CSICFGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01C</td>
<td>RCC_CFGR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x020</td>
<td>RCC_CFGR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x028</td>
<td>RCC_PLL1CFGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x02C</td>
<td>RCC_PLL2CFGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x030</td>
<td>RCC_PLL3CFGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>PLL1R[6:0]</td>
<td>PLL1Q[6:0]</td>
<td>PLL1P[6:0]</td>
<td>PLL1N[8:0]</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>0x034</td>
<td>RCC_PLL1DIVR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 1</td>
<td>0 0 0 0 0 0 1</td>
<td>0 0 0 0 0 0 1</td>
<td>0 1 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x038</td>
<td>RCC_PLL1FRACR</td>
<td></td>
<td>PLL1FRACN[12:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x03C</td>
<td>RCC_PLL2DIVR</td>
<td>PLL2R[6:0]</td>
<td>PLL2Q[6:0]</td>
<td>PLL2P[6:0]</td>
<td>PLL2N[8:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 1</td>
<td>0 0 0 0 0 0 1</td>
<td>0 0 0 0 0 0 1</td>
<td>0 1 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x040</td>
<td>RCC_PLL2FRACR</td>
<td>PLL2FRACN[12:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x044</td>
<td>RCC_PLL3DIVR</td>
<td>PLL3R[6:0]</td>
<td>PLL3Q[6:0]</td>
<td>PLL3P[6:0]</td>
<td>PLL3N[8:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 1</td>
<td>0 0 0 0 0 0 1</td>
<td>0 0 0 0 0 0 1</td>
<td>0 1 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x048</td>
<td>RCC_PLL3FRACR</td>
<td>PLL3FRACN[12:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x050</td>
<td>RCC_CIER</td>
<td>PLL3RDYIE</td>
<td>PLL2RDYIE</td>
<td>PLL1RDYIE</td>
<td>HSI48RDYIE</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x054</td>
<td>RCC_CIFR</td>
<td>PLL3RDYF</td>
<td>PLL2RDYF</td>
<td>PLL1RDYF</td>
<td>HSI48RDYF</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x056</td>
<td>RCC_CICR</td>
<td>PLL3RDYC</td>
<td>PLL2RDYC</td>
<td>PLL1RDYC</td>
<td>HSI48RDYC</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 108. RCC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x060</td>
<td>RCC_AHB1RSTR</td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x064</td>
<td>RCC_AHB2RSTR</td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x06C</td>
<td>RCC_AHB4RSTR</td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x074</td>
<td>RCC_APB1LRSTR</td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x078</td>
<td>RCC_APB1HRSTR</td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x07C</td>
<td>RCC_APB2RSTR</td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x080</td>
<td>RCC_APB3RSTR</td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x084</td>
<td>RCC_AHB1ENR</td>
</tr>
<tr>
<td>Reset value</td>
<td>1 1 1 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x088</td>
<td>RCC_AHB2ENR</td>
</tr>
<tr>
<td>Reset value</td>
<td>1 1 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>
Table 108. RCC register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x094  | RCC_AHB4ENR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value                                      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x09C  | RCC_APB1LENR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value                                     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0A0  | RCC_APB1HENR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value                                     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0A4  | RCC_APB2ENR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value                                     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0A8  | RCC_APB3ENR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value                                     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0B0  | RCC_AHB1LPENR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value                                     | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x0B4  | RCC_AHB2LPENR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value                                     | 1  | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0BC  | RCC_AHB4LPENR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value                                     | 1  | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C4  | RCC_APB1LPENR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value                                     | 1  | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Reset and clock control (RCC) RM0481
| Offset | Register name     | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x0C8  | RCC_APB1HLPMENR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0CC  | RCC_APB2HLPMENR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0D0  | RCC_APB3HLPMENR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0D8  | RCC_CCIPR1       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0DC  | RCC_CCIPR2       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0E0  | RCC_CCIPR3       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0E4  | RCC_CCIPR4       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0E8  | RCC_CCIPR5       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0F0  | RCC_BDCR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Refer to Section 2.3 for the register boundary addresses.
12 Clock recovery system (CRS)

12.1 Introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI48. The CRS provides powerful means to evaluate the oscillator output frequency, based on comparison with a selectable synchronization signal. The CRS is capable of automatic trimming adjustments based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In this case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, sent by a USB host at 1 ms intervals.

The synchronization signal can also be derived from the LSE oscillator output, or generated by user software.

12.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity:
  - LSE oscillator output
  - USB SOF packet reception
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster startup convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

12.3 CRS implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>CRS1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIM width</td>
<td>6 bits</td>
</tr>
</tbody>
</table>
12.4 CRS functional description

12.4.1 CRS block diagram

The CRS synchronization (SYNC) source, selectable through the CRS_CFRGR register, can be the signal from the LSE clock or the USB SOF signal. For better robustness of the SYNC
input, a simple digital filter (2 out of 3 majority votes, sampled by the HSI48 clock) is implemented to filter out glitches. This source signal has a configurable polarity, and can be divided by a programmable binary prescaler, to obtain a synchronization signal in a suitable frequency range (usually around 1 kHz).

For more information on the CRS synchronization source configuration, refer to Section 12.7.2.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

### 12.4.3 Frequency error measurement

The frequency error counter is a 16-bit down/up counter, reloaded with the RELOAD value on each SYNC event. It starts counting down until it reaches the 0 value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit, where it eventually stops (if no SYNC event is received), and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the down-counting phase (before reaching the 0 value), it means that the actual frequency is lower than the target (the TRIM value must be incremented). When it is detected during the up-counting phase, it means that the actual frequency is higher (the TRIM value must be decremented).
12.4.4 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:
- TOLERANCE LIMIT, given directly in the FELIM field of the CRS_CFGR register
- WARNING LIMIT, defined as 3 × FELIM value
- OUTRANGE (error limit), defined as 128 × FELIM value

The result of this comparison is used to generate the status indication and also to control the automatic trimming which is enabled by setting the AUTOTRIMEN bit in the CRS_CR register:
- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one, hence no trimming action is needed.
  - SYNCOK status indicated
  - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
  - SYNCOK status indicated
  - TRIM value adjusted by one trimming step in AUTOTRIM mode
• When the frequency error is above or equal to the warning limit but below the error limit, a stronger trimming action is necessary, and there is a risk that the optimal TRIM value is not reached for the next period.
  – SYNCWARN status indicated
  – TRIM value adjusted by two trimming steps in AUTOTRIM mode
• When the frequency error is above or equal to the error limit, the frequency is out of the trimming range. This can also happen when the SYNC input is not clean, or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
  – SYNCERR or SYNCMISS status indicated
  – TRIM value not changed in AUTOTRIM mode

Note: If the actual value of the TRIM field is close to its limits and the automatic trimming can force it to overflow or underflow, the TRIM value is set to the limit, and the TRIMOVF status is indicated.

In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS_CR register) the TRIM field of CRS_CR is adjusted by hardware and is read-only.

12.4.5 CRS initialization and configuration

RELOAD value

The RELOAD value must be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. This value is decreased by 1, to reach the expected synchronization on the 0 value. The formula is the following:

\[
\text{RELOAD} = \left(\frac{f_{\text{TARGET}}}{f_{\text{SYNC}}} \right) - 1
\]

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

FELIM value

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of oscillator clock ticks. The following formula can be used:

\[
\text{FELIM} = \left(\frac{f_{\text{TARGET}}}{f_{\text{SYNC}}} \right) \times \text{STEP}[\%] / 100\% / 2
\]

The result must be always rounded up to the nearest integer value to obtain the best trimming response. If frequent trimming actions are not needed in the application, the hysteresis can be increased by slightly increasing the FELIM value.

The reset value of the FELIM field corresponds to \(f_{\text{TARGET}} / f_{\text{SYNC}} = 48000\), and to a typical trimming step size of 0.14%.

Note: The trimming step size depends upon the product, check the datasheet for accurate setting.

Caution: There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields, this can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than 128 * FELIM value (OUTRANGE limit).
12.5 **CRS low-power modes**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. CRS interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>CRS registers are frozen. The CRS stops operating until the Stop mode is exited and the HSI48 oscillator is restarted.</td>
</tr>
<tr>
<td>Standby</td>
<td>The CRS peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

Table 111. Effect of low-power modes on CRS

12.6 **CRS interrupts**

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Clear flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected synchronization</td>
<td>ESYNCF</td>
<td>ESYNCE</td>
<td>ESYNCC</td>
</tr>
<tr>
<td>Synchronization OK</td>
<td>SYNCOKF</td>
<td>SYNCokie</td>
<td>SYNCOKC</td>
</tr>
<tr>
<td>Synchronization warning</td>
<td>SYNCWARNF</td>
<td>SYNCWARNIE</td>
<td>SYNCWARNC</td>
</tr>
<tr>
<td>Synchronization or trimming error (TRIMOVF, SYMCSISS, SYNCERR)</td>
<td>ERRF</td>
<td>ERRIE</td>
<td>ERRRC</td>
</tr>
</tbody>
</table>
12.7 CRS registers

Refer to Section 1.2 on page 101 for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

12.7.1 CRS control register (CRS_CR)

Address offset: 0x00
Reset value: 0x0000 2000
Reset value: 0x0000 4000 (products supporting 7-bit TRIM width)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res.</td>
<td>Res.</td>
<td>TRIM[5:0]</td>
<td>SWSYNC</td>
<td>AUTOTRIMEN</td>
<td>CEN</td>
<td>Res.</td>
<td>ESYNCIE</td>
<td>ERRIE</td>
<td>SYNCWARNIE</td>
<td>SYNCKOIE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 TRIM[5:0]: HSI48 oscillator smooth trimming
These bits provide a user-programmable trimming value to the HSI48 oscillator. They can be programmed to adjust to variations in voltage and temperature that influence the oscillator frequency.

The default value is 32, corresponding to the middle of the trimming interval. The trimming step is specified in the product datasheet. A higher TRIM value corresponds to a higher output frequency.

When the AUTOTRIMEN bit is set, this field is controlled by hardware and is read-only.

Bit 7 SWSYNC: Generate software SYNC event
This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.
0: No action
1: A software SYNC event is generated.

Bit 6 AUTOTRIMEN: Automatic trimming enable
This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to Section 12.4.4 for more details.
0: Automatic trimming disabled, TRIM bits can be adjusted by the user.
1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

Bit 5 CEN: Frequency error counter enable
This bit enables the oscillator clock for the frequency error counter.
0: Frequency error counter disabled
1: Frequency error counter enabled
When this bit is set, the CRS_CFGR register is write-protected and cannot be modified.

Bit 4 Reserved, must be kept at reset value.
12.7.2 CRS configuration register (CRS_CFRG)

This register can be written only when the frequency error counter is disabled (CEN bit is cleared in CRS_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

| Bit 30 | SYNCSRC|1:0| | Bit 27 | RELOAD|15:0| |
|--------|--------|--------|--------|--------|--------| |
| rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     |
| 15     | 14     | 13     | 12     | 11     | 10     | 9      | 8      |
| 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
|        |        |        |        |        |        |        |        |

Bit 31 **SYNCPOL**: SYNCP polarity selection

This bit is set and cleared by software to select the input polarity for the SYNC signal source.

0: SYNC active on rising edge (default)
1: SYNC active on falling edge

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **SYNC[1:0]**: SYNC signal source selection

These bits are set and cleared by software to select the SYNC signal source (see Table 110):

00: crs_sync_in_1 selected as SYNC signal source
01: crs_sync_in_2 selected as SYNC signal source
10: crs_sync_in_3 selected as SYNC signal source
11: crs_sync_in_4 selected as SYNC signal source

Note: When using USB LPM (Link Power Management) and the device is in Sleep mode, the periodic USB SOF is not generated by the host. No SYNC signal is therefore provided to the CRS to calibrate the HSI48 oscillator on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE or reference clock on the GPIOs must be used as SYNC signal.

Bit 27 Reserved, must be kept at reset value.
Bits 26:24 **SYNCDIV[2:0]:** SYNC divider

These bits are set and cleared by software to control the division factor of the SYNC signal.

- 000: SYNC not divided (default)
- 001: SYNC divided by 2
- 010: SYNC divided by 4
- 011: SYNC divided by 8
- 100: SYNC divided by 16
- 101: SYNC divided by 32
- 110: SYNC divided by 64
- 111: SYNC divided by 128

Bits 23:16 **FELIM[7:0]:** Frequency error limit

FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS_ISR register. Refer to Section 12.4.4 for more details about FECAP evaluation.

Bits 15:0 **RELOAD[15:0]:** Counter reload value

RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to Section 12.4.3 for more details about counter behavior.

### 12.7.3 CRS interrupt and status register (CRS_ISR)

Address offset: 0x08
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-16</th>
<th>FECAP[15:0]: Frequency error capture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 15</td>
<td>FEDIR: Frequency error direction</td>
</tr>
<tr>
<td>Bit 14-11</td>
<td>Reserved, must be kept at reset value</td>
</tr>
</tbody>
</table>

**Bits 31:16 FECAP[15:0]:** Frequency error capture

FECAP is the frequency error counter value latched in the time of the last SYNC event. Refer to Section 12.4.4 for more details about FECAP usage.

**Bit 15 FEDIR:** Frequency error direction

FEDIR is the counting direction of the frequency error counter latched in the time of the last SYNC event. It shows whether the actual frequency is below or above the target.

- 0: Up-counting direction, the actual frequency is above the target
- 1: Down-counting direction, the actual frequency is below the target

**Bits 14:11 Reserved, must be kept at reset value.**

**Bit 10 TRIMOVF:** Trimming overflow or underflow

This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

- 0: No trimming error signaled
- 1: Trimming error signaled
Bit 9 **SYNCMISS**: SYNC missed

This flag is set by hardware when the frequency error counter reaches value FELIM * 128 and no SYNC is detected, meaning either that a SYNC pulse was missed, or the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, hence some other action must be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC), and an interrupt is generated if the ERRIE bit is set in the CRS.CR register. It is cleared by software by setting the ERRC bit in the CRS.ICR register.

0: No SYNC missed error signaled
1: SYNC missed error signaled

Bit 8 **SYNCERR**: SYNC error

This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to FELIM * 128. This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action has to be taken. An interrupt is generated if the ERRIE bit is set in the CRS.CR register. It is cleared by software by setting the ERRC bit in the CRS.ICR register.

0: No SYNC error signaled
1: SYNC error signaled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCF**: Expected SYNC flag

This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCE bit is set in the CRS.CR register. It is cleared by software by setting the ESYNCC bit in the CRS.ICR register.

0: No expected SYNC signaled
1: Expected SYNC signaled

Bit 2 **ERRF**: Error flag

This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS.CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS.ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.

0: No synchronization or trimming error signaled
1: Synchronization or trimming error signaled

Bit 1 **SYNCWARNF**: SYNC warning flag

This flag is set by hardware when the measured frequency error is greater than or equal to FELIM * 3, but smaller than FELIM * 128. This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS.CR register. It is cleared by software by setting the SYNCWARNC bit in the CRS.ICR register.

0: No SYNC warning signaled
1: SYNC warning signaled

Bit 0 **SYNCOKF**: SYNC event OK flag

This flag is set by hardware when the measured frequency error is smaller than FELIM * 3. This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS.CR register. It is cleared by software by setting the SYNCOKKC bit in the CRS.ICR register.

0: No SYNC event OK signaled
1: SYNC event OK signaled
### 12.7.4 CRS interrupt flag clear register (CRS_ICR)

Address offset: 0x0C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Bits 31:4</th>
<th>Bits 31:4 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bit 3</td>
<td>ESYNCC: Expected SYNC clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the ESYNCF flag in the CRS_ISR register.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit 2</td>
<td>ERRC: Error clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears TRIMOVF, SYNCMISS and SYNCERR bits and consequently also the ERRF flag in the CRS_ISR register.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit 1</td>
<td>SYNCWARNC: SYNC warning clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the SYNCWARNF flag in the CRS_ISR register.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit 0</td>
<td>SYNCOKC: SYNC event OK clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the SYNCOKF flag in the CRS_ISR register.</td>
<td></td>
</tr>
</tbody>
</table>

### 12.7.5 CRS register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Bits 31:4</th>
<th>Bits 31:4 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>CRS_CR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>CRS_CFGR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>CRS_ISR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Table 113. CRS register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Bits 31:4</th>
<th>Bits 31:4 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>CRS_CR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>CRS_CFGR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>CRS_ISR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Refer to *Section 2.3 on page 111* for the register boundary addresses.

| Offset | Register | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x0C   | CRS_ICR  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| Reset value |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
13 General-purpose I/Os (GPIO)

13.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 16 bits reset register (GPIOx_BRR) and a 32-bit set/reset register (GPIOx_BSRR).

In addition, all GPIOs have a 32-bit locking register (GPIOx_LCKR), two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL), a secure configuration register (GPIOx_SECCFGR) and a high-speed low-voltage register (GPIOx_HSLVR).

13.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_BSRR) for bitwise write access to GPIOx_ODR
- Lock mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions
- TrustZone security support
- I/Os state retention during Standby mode

13.3 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability
Each I/O port bit is freely programmable, however the I/O port registers must be accessed as 32-bit words, half-words or bytes. The GPIOx BSRR and GPIOx BRR registers allow atomic read/modify accesses to any of the GPIOx ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

The figure below shows the basic structure of a three-volt or five-volt tolerant GPIO (TT or FT). The Table 114 gives the possible port bit configurations.

**Figure 55. Structure of three-volt or five-volt tolerant GPIO (TT or FT)**

![Figure 55. Structure of three-volt or five-volt tolerant GPIO (TT or FT)](image)

**Note:** On a TT GPIO, the analog switch is not present and replaced by a direct connection. The analog bloc parasitic circuitry does not allow five-volt tolerance.

**Table 114. Port bit configuration**

<table>
<thead>
<tr>
<th>MODE(i) [1:0]</th>
<th>OTYPE(i)</th>
<th>OSPEED(i) [1:0]</th>
<th>PUPD(i) [1:0]</th>
<th>I/O configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>GP output PP</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>GP output PP + PU</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>GP output PP + PD</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>GP output OD</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>GP output OD + PU</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Reserved (GP output OD)</td>
</tr>
</tbody>
</table>
13.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in analog mode.

The debug pins are in AF pull-up/pull-down after reset:
- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDIO in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO/TRACESWO in floating state no pull-up/pull-down

BOOT0 is in input mode during the reset until at least the end of the option byte loading phase (see Section 13.3.15: I/Os state retention during standby mode).

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, that can be activated or not depending on the value in the GPIOx_PUPDR register.
13.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there is no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to 16 alternate function inputs (AF0 to AF15) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:

- After reset, the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user must proceed as follows:

- **Debug function**: after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host.
- **GPIO**: configure the desired I/O as output, input or analog in the GPIOx_MODER register.
- **Peripheral alternate function**:
  - Connect the I/O to the desired AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDR registers respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- **Additional functions**:
  - For the ADC and DAC configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC and DAC registers.
  - For the additional functions like RTC, WKUPx and oscillators, configure the required function in the related RTC, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

13.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.
13.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIO port input data register (GPIOx_IDR) \((x = \text{A to I})\) and GPIO port output data register (GPIOx_ODR) \((x = \text{A to I})\)).

GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.

13.3.5 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register that allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, BS(i) sets the corresponding ODR(i) bit. When written to 1, BR(i) resets the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: one or more bits can be modified in a single atomic AHB write access.

13.3.6 GPIO locking mechanism

The GPIO control registers can be frozen by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, GPIOx_AFRH and GPIOx_HSLVR.

To write the GPIOx_LCKR register, a specific write/read sequence must be applied. When the right LOCK sequence is applied to the bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence is applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH).

The LOCK sequence can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 must be set at the same time as the [15:0] bits.

13.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can
thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the device datasheet.

### 13.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port can be configured in input, output or alternate function mode (the port must not be configured in analog mode). Refer to *Section 23: Extended interrupts and event controller (EXTI)*.

### 13.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register provides the I/O state.

The figure below shows the input configuration of the I/O port bit.

**Figure 56. Input floating/pull-up/pull-down configurations**
13.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open-drain mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register leaves the port in Hi-Z (the P-MOS is never activated).
  - Push-pull mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register activates the P-MOS.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.
- A read access to the output data register gets the last written value.

The figure below shows the output configuration of the I/O port bit.

![Figure 57. Output configuration](image)

13.3.11 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode.
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data).
- The Schmitt trigger input is activated.
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.
The figure below shows the alternate function configuration of the I/O port bit.

**Figure 58. Alternate function configuration**

**13.3.12 Analog configuration**

When the I/O port is programmed as analog configuration:
- The output buffer is disabled.
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up and pull-down resistors are disabled by hardware.
- Read access to the input data register gets the value 0.

The figure below shows the high-impedance, analog-input configuration of the I/O port bits.

**Figure 59. High-impedance analog configuration**

**13.3.13 Using the HSE or LSE oscillator pins as GPIOs**

When the HSE or LSE oscillator is switched off (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched on (by setting the HSEON or LSEON bit in the RCC_CSR register), the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.
When the oscillator is configured in a user external clock mode, only the pin is reserved for clock input, and the OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

### 13.3.14 Using the GPIO pins in the RTC supply domain

The PC13/PC14/PC15/PI8 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

For details about I/O control by the RTC, refer to Section 46.3: RTC functional description.

### 13.3.15 I/Os state retention during standby mode

In the Standby mode, the I/Os are by default in floating state.

If the IORETEN bit in the PWR_IORETR register is set, the I/Os state is sampled during standby entry. The state of I/Os is applied to the pin via pull-up and pull-down resistors. The pull-up and pull-down resistors remains applied after Standby wakeup until the IORETEN bit in the PWR_IORETR register is cleared by software.

### 13.3.16 TrustZone security

The TrustZone security is activated by the TZEN option byte in the Flash Option Byte register. When the TrustZone is active (TZEN = 0xB4), each I/O pin of GPIO port can be individually configured as secure through the GPIOx_SECCFGR register.

When the selected I/O pin is configured as secure, its corresponding configuration bits for alternate function, mode selection, I/O data are secure against a non-secure access. In case of non-secure access, these bits are RAZ/WI.

The I/Os with peripherals functions are also conditioned by the peripheral security configuration (see Section 5: Global TrustZone controller (GTZC) for more details):

- For peripherals for which the I/O pin selection is done through alternate functions registers: if the peripheral is configured as secure, it cannot be connected to a non-secure I/O pin. If this is not respected, the input data to the secure peripheral is forced to 0 (I/O input pin value is ignored) and the output pin value is forced to 0, thus avoiding any secure information leak through non-secure I/Os.

- For I/Os with analog switches, directly controlled by peripherals (such as ADC for instance): If the I/O is secure, the I/O analog switch cannot be controlled by a non-secure peripheral. If this is not respected, the switch remains open. This prevent the redirection of secure data to a non-secure peripheral or I/O through analog path. Refer to Section 3: System security for more details.

- Some of the paths between I/Os “additional functions” and peripherals are not blocked if the I/O is secure and the peripheral is non-secure. Therefore it is recommended to configure those peripherals as secure even when not used by the application. Refer to Section 3: System security for the list of concerned peripherals. When the path has a security control, it follows the same rule as I/O selection through alternate functions.

Refer to the device pins definition table in datasheet for more information about peripherals alternate functions and additional functions mapping.

After reset, all GPIO ports are secure.
The table below gives a summary of the I/O port secured bits following the security configuration bit in the GPIO_SECCFGR register. When the I/O bit port is configured as secure:

- **Secured bits:** read and write operations are only allowed by a secure access. Non-secure read or write accesses on secured bits are RAZ/WI. There is no illegal access event generated.
- **Non-secure bits:** no restriction. Read and write operations are allowed by both secure and non-secure accesses.

When the TrustZone security is disabled (TZEN = 0xC3 in FLASH_OPTSR2 register), all registers bits are non-secure. The GPIOx_SECCFGR register is RAZ/WI.

### Table 115. GPIO secured bits

<table>
<thead>
<tr>
<th>Secure configuration bit</th>
<th>Secured bit</th>
<th>Register name</th>
<th>Non-secure access on secure bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECy = 1 in GPIOx_SECCFGR(^{(1)})</td>
<td>MODEy[1:0]</td>
<td>GPIOx_MODER</td>
<td>RAZ/WI</td>
</tr>
<tr>
<td></td>
<td>OTy</td>
<td>GPIOx_OTYPER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OSPEEDy[1:0]</td>
<td>GPIOx_OSPEEDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PUPDy[1:0]</td>
<td>GPIOx_PUPDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IDy</td>
<td>GPIOx_IDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ODy</td>
<td>GPIOx_ODR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BSy and BRy</td>
<td>GPIOx_BSR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LCKy</td>
<td>GPIOx_LCKR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BRy</td>
<td>GPIOx_BRR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AFSELy[3:0]</td>
<td>GPIOx_AFRH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HSLVy</td>
<td>GPIOx_HSLVR</td>
<td></td>
</tr>
</tbody>
</table>

1. GPIOx, x = A to I. For x = A to H, y = 0 to 15. For x = I, y = 0 to 11.

### 13.3.17 Privileged and unprivileged modes

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

### 13.3.18 High-speed low-voltage mode (HSLV)

Some I/Os have the capability to increase their maximum speed at low voltage by configuring them in HSLV mode. The I/O HSLV bit controls whether the I/O output speed is optimized to operate at 3.3 V (default setting) or at 1.8 V (HSLV = 1).

**Caution:** The I/O HSLV configuration bit must not be set if the I/O supply (V\(_{DD}\) or V\(_{DDIO2}\)) is above 2.7 V. Setting it while the voltage is higher than 2.7 V can damage the device. The I/O HSLV bit can be set only when the corresponding option bit is activated (IO_VDD_HSLV or IO_VDDIO2_HSLV depending on the I/O supply, refer to Section 7.4: FLASH option bytes). There is no hardware protection associated to this feature so it is recommended to use it only as a static configuration for fixed I/O supply.
13.3.19 I/O compensation cell

The I/O commutation slew rate (tfall / trise) can be adapted by software depending on process, voltage and temperatures conditions, in order to reduce the I/O noise on power supply. Refer to Section 14: System configuration, boot, and security (SBS) for more details.

13.4 GPIO registers

This section gives a detailed description of the GPIO registers.

The peripheral registers can be written in word, half word or byte mode.

13.4.1 GPIO port mode register (GPIOx_MODER) (x = A to I)

Address offset: 0x00
Reset value: 0xABFF FFFF (for port A)
Reset value: 0xFFFF FEBF (for port B)
Reset value: 0xFFFF FFFF (for ports C..H)
Reset value: 0x00FF FFFF (for port I)

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

Bits 31:0 MODEy[1:0]: Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to I)

Address offset: 0x04
Reset value: 0x0000 0000
13.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to I)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)
Reset value: 0x0000 00C0 (for port B)
Reset value: 0x0000 0000 (for the other ports)

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Bit Description</th>
<th>Bit Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>15:0</td>
<td>OTy</td>
<td>Port x configuration I/O pin y (y = 15 to 0)</td>
</tr>
<tr>
<td></td>
<td>These bits are written by software to configure the I/O output type.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Output push-pull (reset state)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Output open-drain</td>
<td></td>
</tr>
</tbody>
</table>

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

<table>
<thead>
<tr>
<th>Bit 31:0</th>
<th>OSPEEDy[1:0]</th>
<th>Port x configuration I/O pin y (y = 15 to 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>These bits are written by software to configure the I/O output speed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00: Low speed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>01: Medium speed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10: High speed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11: Very-high speed</td>
<td></td>
</tr>
</tbody>
</table>

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to I)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)
Reset value: 0x0000 0100 (for port B)
Reset value: 0x0000 0000 (for the other ports)

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Bit Description</th>
<th>Bit Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>PUPDy[1:0]</td>
<td>Port x configuration I/O pin y (y = 15 to 0)</td>
</tr>
<tr>
<td></td>
<td>These bits are written by software to configure the pull-up/pull-down function.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00: Pull-up</td>
<td></td>
</tr>
<tr>
<td></td>
<td>01: Pull-down</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10: Pull-up</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11: Pull-down</td>
<td></td>
</tr>
</tbody>
</table>

Note: Refer to the device datasheet for the power supply and load conditions for each function.

The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.
13.4.5 GPIO port input data register (GPIOx_IDR) (x = A to I)

Address offset: 0x10
Reset value: 0x0000 XXXX

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 IDy: Port x input data I/O pin y (y = 15 to 0)
These bits are read-only. They contain the input value of the corresponding I/O port.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.6 GPIO port output data register (GPIOx_ODR) (x = A to I)

Address offset: 0x14
Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 ODy: Port output data I/O pin y (y = 15 to 0)
These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A to I).
The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.
13.4.7  GPIO port bit set/reset register (GPIOx_BSRR) (x = A to I)

Address offset: 0x18
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>BR15</th>
<th>BR14</th>
<th>BR13</th>
<th>BR12</th>
<th>BR11</th>
<th>BR10</th>
<th>BR9</th>
<th>BR8</th>
<th>BR7</th>
<th>BR6</th>
<th>BR5</th>
<th>BR4</th>
<th>BR3</th>
<th>BR2</th>
<th>BR1</th>
<th>BR0</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
BS15 BS14 BS13 BS12 BS11 BS10 BS9 BS8 BS7 BS6 BS5 BS4 BS3 BS2 BS1 BS0
| w    | w    | w    | w    | w    | w    | w    | w    | w    | w    | w    | w    | w    | w    | w    | w    |

Bits 31:16  BRY: Port x reset I/O pin y (y = 15 to 0)
- These bits are write-only. A read to these bits returns the value 0x0000.
- 0: No action on the corresponding ODy bit
- 1: Resets the corresponding ODy bit

Note: If both BSy and BRy are set, BSy has priority.
The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bits 15:0  BSy: Port x set I/O pin y (y = 15 to 0)
- These bits are write-only. A read to these bits returns the value 0x0000.
- 0: No action on the corresponding ODy bit
- 1: Sets the corresponding ODy bit

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.8  GPIO port configuration lock register (GPIOx_LCKR) (x = A to I)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C
Reset value: 0x0000 0000

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

15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
LCK15 LCK14 LCK13 LCK12 LCK11 LCK10 LCK9 LCK8 LCK7 LCK6 LCK5 LCK4 LCK3 LCK2 LCK1 LCK0
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
Bits 31:17  Reserved, must be kept at reset value.

Bit 16  **LCKK**: Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active
1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

- **LOCK key write sequence**:
  WR LCKR[16] = 1 + LCKR[15:0]
  WR LCKR[16] = 0 + LCKR[15:0]
  WR LCKR[16] = 1 + LCKR[15:0]

- **LOCK key read**
  RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)

*Note: During the LOCK key write sequence, the value of LCK[15:0] must not change. Any error in the lock sequence aborts the LOCK.*

After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:0  **LCKy**: Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is 0
0: Port configuration not locked
1: Port configuration locked

*Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.*

### 13.4.9  GPIO alternate function low register (GPIOx_AFRL) (x = A to I)

Address offset: 0x20

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Bits 31:0 **AFSELy[3:0]**: Alternate function selection for port x I/O pin y (y = 7 to 0)

These bits are written by software to configure alternate function I/Os.
- 0000: AF0
- 0001: AF1
- 0010: AF2
- 0011: AF3
- 0100: AF4
- 0101: AF5
- 0110: AF6
- 0111: AF7
- 1000: AF8
- 1001: AF9
- 1010: AF10
- 1011: AF11
- 1100: AF12
- 1101: AF13
- 1110: AF14
- 1111: AF15

*Note:* The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

### 13.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to H)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

<table>
<thead>
<tr>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:0  **AFSELy[3:0]**: Alternate function selection for port x I/O pin y (y = 15 to 8)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

*Note:* The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

### 13.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to I)
Address offset: 0x28
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>BR15</td>
<td>BR14</td>
<td>BR13</td>
<td>BR12</td>
<td>BR11</td>
<td>BR10</td>
<td>BR9</td>
<td>BR8</td>
<td>BR7</td>
<td>BR6</td>
<td>BR5</td>
<td>BR4</td>
<td>BR3</td>
<td>BR2</td>
<td>BR1</td>
<td>BR0</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:16  **Reserved**, must be kept at reset value.

Bits 15:0  **BRy**: Port x reset I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODy bit
1: Reset the corresponding ODy bit

*Note:* The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.
13.4.12 GPIO high-speed low-voltage register (GPIOx_HSLVR) (x = A to I)

Address offset: 0x2C
Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 HSLV: Port x high-speed low-voltage configuration (y = 15 to 0)

These bits are written by software to optimize the I/O speed when the I/O supply is low. Each bit is active only if the corresponding IO_VDD_HSLV/IO_VDDIO2_HSLV user option bit is set. It must be used only if the I/O supply voltage is below 2.7 V. Setting these bits when the I/O supply (VDD or VDDIO2) is higher than 2.7 V may be destructive.

0: I/O speed optimization disabled
1: I/O speed optimization enabled

Note: Not all I/Os support the HSLV mode. Refer to the I/O structure in the corresponding datasheet for the list of I/Os supporting this feature. Other I/Os HSLV configuration must be kept at reset value.

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.13 GPIO secure configuration register (GPIOx_SECCFGR) (x = A to I)

When the system is secure (TZEN = 0xB4), this register provides write access security and can be written only by a secure access. It is used to configure a selected I/O as secure. A non-secure write access to this register is discarded.

When the system is not secure (TZEN = 0xC3), this register is RAZ/WI.

Address offset: 0x30
Reset value: 0x0000 FFFF (for ports A to H)
Reset value: 0x0000 0FFF (for port I)
### 13.4.14 GPIO register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Bits 31:16</th>
<th>Bits 15:0 SECy: I/O pin of Port x secure bit enable y (y = 15 to 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>GPIOx_MODER (x = A to I)</td>
<td>Reserved</td>
<td>Secure bit enable y (y = 15 to 0)</td>
</tr>
<tr>
<td>0x04</td>
<td>GPIOx_OTYPER (x = A to I)</td>
<td>Reserved</td>
<td>Secure bit enable y (y = 15 to 0)</td>
</tr>
<tr>
<td>0x08</td>
<td>GPIOx_OSPEEDR (x = A to I)</td>
<td>Reserved</td>
<td>Secure bit enable y (y = 15 to 0)</td>
</tr>
<tr>
<td>0x0C</td>
<td>GPIOx_PUPDR (x = A to I)</td>
<td>Reserved</td>
<td>Secure bit enable y (y = 15 to 0)</td>
</tr>
<tr>
<td>0x10</td>
<td>GPIOx_IDR (x = A to I)</td>
<td>Reserved</td>
<td>Secure bit enable y (y = 15 to 0)</td>
</tr>
<tr>
<td>0x14</td>
<td>GPIOx_ODRR (x = A to I)</td>
<td>Reserved</td>
<td>Secure bit enable y (y = 15 to 0)</td>
</tr>
<tr>
<td>0x18</td>
<td>GPIOx_BSRR (x = A to I)</td>
<td>Reserved</td>
<td>Secure bit enable y (y = 15 to 0)</td>
</tr>
<tr>
<td>0x1C</td>
<td>GPIOx_LCKR (x = A to I)</td>
<td>Reserved</td>
<td>Secure bit enable y (y = 15 to 0)</td>
</tr>
</tbody>
</table>

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.
Refer to Section 2.3 for the register boundary addresses.
14 System configuration, boot, and security (SBS)

14.1 SBS introduction

The STM32H563/H573 and STM32H562 devices feature a set of configuration registers located in the SBS. On top of various device configurations, this SBS peripheral controls key boot and security features, including debug control and secure storage control.

14.2 SBS main features

- System configuration
  - Manage safety feature
  - Enable/disable the FMP (Fast-mode Plus) high-drive capability of some I/Os and voltage booster for I/O analog switches
  - Manage the I/O compensation cell
  - Configure register security access

- Boot control
  - Upon system reset, configure the Cortex-M33 boot address and the temporal isolation level depending on the current configuration such as PRODUCT_STATE (sbs_product_state), BOOT_UBE (sbs_irot_select), or TZEN (sbs_tzen).
  - Manage the temporal isolation feature implemented thanks to the hide protect level (HDPL) monotonic counter

- Debug control
  - Control the opening of the device debug interface, ensuring the sequencing of events that guarantee the device security

- Hardware secure storage control
  - Control the secure storage selections: OBK-HDPL selection (selects the area of secure storage of the flash memory and selects the corresponding DHUK in SAES), EPOCH selection
14.3 SBS functional description

14.3.1 SBS block diagram

The figure below shows the SBS block diagram, including the four main functions:
- System configuration
- Boot control
- Debug control
- Hardware secure storage control

Figure 60. SBS block diagram
14.3.2 SBS signals

The table below details the user relevant internal signals that interface the SBS.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOT0</td>
<td>Input</td>
<td>Select booting on user flash memory or Bootloader when TrustZone is disabled (TZEN = 0xC3), or on RSS when TrustZone is enabled (TZEN = 0xB4).</td>
</tr>
<tr>
<td>sbs_irot_select</td>
<td>Signal</td>
<td>Signal based on BOOT_UBE option byte to select the iROT between ST-iROT and user flash memory (OEM-iROT)</td>
</tr>
<tr>
<td>sbs_tzen</td>
<td>Signal</td>
<td>Signal based on TZEN option byte to activate/deactivate the TrustZone</td>
</tr>
<tr>
<td>sbs_boot_addresses</td>
<td>List of addresses defined by the flash memory:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>– NSBOOTADD: non-secure boot address</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– SECBOOTADD: secure boot address</td>
</tr>
<tr>
<td>sbs_product_state</td>
<td>List of addresses defined by the flash memory:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>– NSBOOTADD: non-secure boot address</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– SECBOOTADD: secure boot address</td>
</tr>
<tr>
<td>sbs_dbg_req</td>
<td>Launch the debug authentication protocol when booting.</td>
<td></td>
</tr>
<tr>
<td>sbs_init_vtor_s</td>
<td>Vector address for Cortex-M33 secure entry point.</td>
<td></td>
</tr>
<tr>
<td>sbs_init_vtor_ns</td>
<td>Vector address for Cortex-M33 non-secure entry point.</td>
<td></td>
</tr>
<tr>
<td>sbs_tz_state</td>
<td>Inform the Cortex-M33 on the secure state of the core.</td>
<td></td>
</tr>
<tr>
<td>sbs_hdpl</td>
<td>Output</td>
<td>HDPL (temporal isolation level, ID of the boot level, used to isolate boot levels) This signal reflects a monotonic counter that can be incremented from 0 to 3, and that is reset to zero only on software reset or POR.</td>
</tr>
<tr>
<td>sbs_ap_unlocked</td>
<td>Control the Cortex-M33 access port.</td>
<td></td>
</tr>
<tr>
<td>sbs_dbg_unlocked</td>
<td>Unlock the debug (when set to 1) for the Cortex-M33 non-secure part.</td>
<td></td>
</tr>
<tr>
<td>sbs_dbg_unlocked_sec</td>
<td>Unlock the debug (when set to 1) for the Cortex-M33 secure part.</td>
<td></td>
</tr>
<tr>
<td>sbs_dbg_reset</td>
<td>Input</td>
<td>This signal is used to control the reset of the debug authentication configuration to be done with a system reset or a power-on reset. The configuration is done through the DBGMCU using the DCRT bitfield of DBGMCU_CR register.</td>
</tr>
<tr>
<td>sbs_obk_hdpl</td>
<td>Output</td>
<td>Select secure storage domain (OBK-HDPL) for current HDPL, or greater ones (to allow provisioning).</td>
</tr>
<tr>
<td>sbs_epoch_out</td>
<td>Output</td>
<td>EPOCH counters (NS_EPOCH and SEC_EPOCH, inputs from flash memory) are used to manage the REPLAY protection.</td>
</tr>
<tr>
<td>sbs_sec_nsec</td>
<td>Reflect secure/non-secure selection for the secure storage.</td>
<td></td>
</tr>
</tbody>
</table>

14.3.3 SBS reset and clocks

The SBS configuration port is clocked by the AHB bus clock. There is a general reset and a debug configuration reset controlled in DBGMCU.
14.3.4 SBS system configuration

SBS I/O compensation cell management

The I/O compensation cell generates an 8-bit value for the I/O buffer (4 bits for N-MOS and 4 bits for P-MOS), that depends on PVT operating conditions (process, voltage, temperature). These bits are used to control the output impedance in the I/O buffer, and the slew rate of the I/O commutation (the $t_{\text{fall}}$ and $t_{\text{rise}}$ time), in order to reduce the I/O noise on power supply.

As shown in the figure below, the compensation cell is split in two blocks: one block to provide an optimal code for the current PVT, and one block to drive the block controlled by the software.

**Figure 61. Compensation cell management**

The compensation cell value can be read when the READY flag is set in SBS_CCCSR. With CODESEL in SBS_CCCSR, the application can select the value to apply between two options: the code from the cell or the code from SBS_CCSWCR.

Two compensation cells are embedded in STM32H563/H573 and STM32H562 devices:
- one for the I/Os supplied by VDDIO power rail
- one for the I/Os supplied by VDDIO2 power rail

By default, the compensation cells are disabled, and a fixed code is applied to all the I/Os.

**Note:** The compensation cell can be used only when $2.7 \, \text{V} \leq V_{\text{DDIOx}} \leq 3.6 \, \text{V}$ or $1.62 \, \text{V} \leq V_{\text{DDIOx}} \leq 2 \, \text{V}$ (see Figure 62).

**Note:** The compensation cell can be used only when the CSI oscillator is enabled, see Section 11: Reset and clock control (RCC) for more details on CSI oscillator.
SBS TrustZone security and privilege

- **SBS TrustZone security**
  
  When the TrustZone security is activated, the SBS is able to protect secure registers from being modified by non-secure accesses.
  
  The TrustZone security is activated by the TZEN option byte in FLASH_OPTSR2_PRG. A non-secure read/write access to a secured register is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the SBS illegal access event is enabled in the GTZC.

- **Privileged/unprivileged mode**
  
  The SBS registers can be read and written by privileged and unprivileged accesses except the SBS registers for CPU configuration:
  
  - SBS_CSLCKR, SBS_FPUIMR and SBS_CNSLCKR
  - FPUSEC in SBS_SECCFGR

  An unprivileged access to a privileged register is RAZ/WI.

14.3.5 **SBS boot control**

The SBS can be used to control the boot entry points considering the product settings. The main boot control actions are listed below:

- Run product with or without TrustZone enabled.
- Select between ST-iROT or OEM-iROT.
- Boot when launching a debug authentication sequence.
- Select boot between the Bootloader or the user flash memory boot.
- Initialize the HDPL boot value.
The boot configurations are selected considering the product settings:

- **BOOT0**: to select booting on user flash memory or RSS (root secure services)
- **BOOT_UBE option byte**: to select the iROT between ST-iROT and OEM-iROT
- **TZEN option byte**: to activate/deactivate the Trust Zone
- **sbs_boot_addresses**: list of addresses defined by the flash memory:
  - **NSBOOTADD**: non-secure boot address
  - **SECBOOTADD**: secure boot address
- **PRODUCT_STATE**: option byte to activate the different security mechanisms depending on the product use. Expected values are described in Section 7: Embedded flash memory (FLASH).
- **sbs_dbg_req**: used to launch the debug authentication protocol when booting

The boot control logic sets the following data:

- **sbs_init_vtor_s**: vector address for Cortex-M33 secure entry point
- **sbs_init_vtor_ns**: vector address for Cortex-M33 non-secure entry point
- **sbs_tz_state (secure/non-secure)**: informs the Cortex-M33 on the secure state of the core.
- **HDPL** (see the description below)

### SBS HDPL (temporal isolation level) management

The HDPL is a monotonic counter incremented during the boot stages. The HDPL is reset to its default value only after a power-on or a system reset. This default value (0 or 1) depends on the device life cycle, as defined in boot logic.

The STM32H563/H573 and STM32H562 devices use HDPL information to automatically isolate code and its associated secrets (like keys) during the boot process. Incrementing HDPL ensures that private code and data for one boot stage cannot be directly accessible from later boot stages.

The HDPL is used by the user flash memory, see Section 7: Embedded flash memory (FLASH) for more details. The HDPL can take values from 0 to 3. When reaching the value 3, HDPL keeps this value until reset. The current HDPL value is readable in HDPL bitfield in SBS_HDPLSR.
To increment the HDPL by one, the application must write 0x6A to INCR_HDPL in SBS_HDPLCR. After such increment, and before doing any subsequent action, the user must check that the HDPL has effectively been incremented reading SBS_HDPLSR.

### Table 118. HDPL encoded values

<table>
<thead>
<tr>
<th>HDPL</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0xB4</td>
</tr>
<tr>
<td>1</td>
<td>0x51</td>
</tr>
<tr>
<td>2</td>
<td>0x8A</td>
</tr>
<tr>
<td>3</td>
<td>0x6F</td>
</tr>
<tr>
<td>All other values</td>
<td></td>
</tr>
</tbody>
</table>
14.3.6 **SBS debug control**

The SBS debug control is used to manage debug opening, taking care on the product context (PRODUCT_STATE, TZEN, HDPL) on register settings, or through a debug authentication control.

When the debug is forbidden, the mailbox access port, Cortex-M33 access port and CPU debug interface are locked. In this situation, the debugger cannot access the CPU and no effective debug can be done. Refer to the [Section 58: Debug support (DBG)](#) for more details.

**Authenticated debug sequence**

1. The external host requests to launch the debug authentication protocol, via the DBGMCU access port mailbox. The rest of the device is kept under reset.
2. SBS selects the STMicroelectronics RSS-DA (debug authentication library) boot address, and requests the CPU to be released from reset.
3. The CPU running RSS-DA library executes the debug authentication protocol in the system flash memory. If the device is closed, the access port mailbox is closed until RSS-DA acknowledges the authentication sequence start request.

4. The authentication method depends on TrustZone activation:
   - When TrustZone is activated (TZEN = 0xB4), the authentication method is based on certificates. As soon as a debug certificate chain is fully verified by the device, if the certificate concerns a debug permission, the RSS-DA programs the debug opening of the Cortex-M33. Alternatively, the certificate can authorize partial or full regression, allowing debug on a regressed part.
   - When TrustZone is disabled (TZEN = 0xC3), the authentication method is based on password. This method only allows the full regression of the product to be controlled.

5. Above reopenings are effective only when HDPL in SBS_HDPLSR has a value equal or superior to the value programmed in DBG_AUTH_HDPL in SBS_DBGCR. In case of authentication failure, the user is informed through the host interface.

Note: The debug authentication library in system flash memory is available only when HDPL = 0 or 1 in SBS_HDPLSR. Only this library can perform the steps 3 and 4 described above.

Debug reset

The debug opening can have the configuration to be reset by system or power-on reset, depending on a DBGMCU register field.

Debug locking

The debug configuration can be locked thanks to DBGCFG_LOCK in SBS_DBGLOCKR. SBS_DBGCR is then no longer writable.

When DBGCFG_LOCK is set to 1, it can be reset only by system or power-on reset. The configuration is done through the DBGMCU using the DCRT field of DBGMCU_CR.

Figure 64. SBS debug control
Inputs used to control the debug opening:

- **sbs_dbg_req**: input signal that a host requests to launch the debug authentication protocol. When this signal is set, the boot address is set to launch the debug authentication library.

- **sbs_tzen**: inform on the selected platform configuration related to TrustZone activation. When TZEN value is set to zero (0xC3), the platform does not support TrustZone and the simplified life cycle is proposed. See Section 7: Embedded flash memory (FLASH) for more details.

- **sbs_product_state**: provide the current PRODUCT_STATE (from flash memory). See Section 7: Embedded flash memory (FLASH) for more details.

- **sbs_dbg_reset**: reset information coming from the RCC, and reset SBS_DBGCR if this register is configured to be reset.

**Configuration**

- **DBG_UNLOCK** in SBS_DBGCR: debug unlock when DBG_AUTH_HDPL is reached

- **AP_UNLOCK** in SBS_DBGCR: access port unlock

- **DBG_AUTH_HDPL** in SBS_DBGCR: authenticated debug temporal isolation level. Define the HDPL value from which the debug can be opened. Value can only be from 1 to 3.

- **DBG_AUTH_SEC** in SBS_DBGCR: specify if the debug reopening is for non-secure only or for both (secure and non-secure).

- **DBGC_LOCK** in SBS_DBGLOCKR: lock the current debug configuration (released only by a reset).

**Outputs**

- **sbs_ap_unlocked**: signal controlling the Cortex-M33 access port
  - 1: 0xB4
  - 0: all other codes

- **sbs_dbg_unlocked**: unlock the debug (when set to 1) for the Cortex-M33 non-secure.
  - 1: 0xB4
  - 0: all other codes

- **sbs_dbg_unlocked_sec**: unlock the debug (when set to 1) for the Cortex-M33 secure.
  - 1: 0xB4
  - 0: all other codes

**14.3.7 SBS hardware secure storage control**

This feature ensures the isolation of keys and data related to ROT (root-of-trust) when re-opening the debug (when product in the field).

This includes a dedicated area called OB-Keys in the flash memory (see Section 7: Embedded flash memory (FLASH) for more details), and the key derivation (DHUK) mechanism protecting data using a hardware key different for the identified domains.
• sbs_obk_hdpl: select secure storage context for current HDPL, or greater ones (to allow provisioning). This signal is set thanks to the SBS_NEXTHDPLCR register. This information is used by the flash memory to select the right area, and by the SAES when processing the derived key to generate a key derived related to HDPL context.

• sbs_epoch_out: EPOCH counters are used to manage the REPLAY protection. These counters are incremented when regressions are done. Injecting the EPOCH in the DHUK (in the SAES) ensures that all information encrypted with the DHUK cannot be replayed after a regression.

• sbs_sec_nsec: apply a different protection DHUK for secure and non-secure assets.
All keys encrypted using the SAES/DHUK inherit of the RHUK property: unique per device.
All data encrypted thanks to the SAES using the DHUK are specific to a combination of [sbs_obk_hdpl + sbs_epoch_out + sbs_sec_nsec].

Inputs used to control the hardware secure storage control
sbs_epoch_in and sbs_epoch_in_ns: 24-bit values coming from the flash memory and representing regression counters respectively for secure and non-secure.

Configuration
• NEXTHDPL in SBS_NEXTHDPLCR: access to next HDPL secure storage areas.
• EPOCH_SEL in SBS_EPOCHSELCR: select the EPOCH source related to the secure storage under control (SEC_EPOCH, NS_EPOCH, or FORCED_To_Zero).
14.4 SBS interrupts

SBS does not support interrupts.

14.5 SBS registers

14.5.1 SBS temporal isolation control register (SBS_HDPLCR)

Address offset: 0x010
Reset value: 0x0000 00B4
Reset: system reset
Register security: no restriction

<table>
<thead>
<tr>
<th>HDPL[7:0] in SBS_HDPLSR</th>
<th>NEXTHDPL[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0x1</td>
</tr>
<tr>
<td>0 (0xB4)</td>
<td>0xB4</td>
</tr>
<tr>
<td>1 (0x51)</td>
<td>0x51</td>
</tr>
<tr>
<td>2 (0x8A)</td>
<td>0x8A</td>
</tr>
<tr>
<td>3 (0x6F)</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td></td>
</tr>
</tbody>
</table>

Table 120. OBK-HDPL logic

<table>
<thead>
<tr>
<th>HDPL[7:0] in SBS_HDPLSR</th>
<th>NEXTHDPL[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0x1</td>
</tr>
<tr>
<td>0 (0xB4)</td>
<td>0xB4</td>
</tr>
<tr>
<td>1 (0x51)</td>
<td>0x51</td>
</tr>
<tr>
<td>2 (0x8A)</td>
<td>0x8A</td>
</tr>
<tr>
<td>3 (0x6F)</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0 **INCR_HDPL[7:0]**: increment HDPL value
- 0xB4: no increment
- 0x6A: recommended value to increment HDPL level by one
- Other: all other values allow a HDPL level increment.
### 14.5.2 SBS temporal isolation status register (SBS_HDPLSR)

Address offset: 0x014  
Reset value: 0xXXXX XXXX  
The reset value depends on boot case: booting with HDPL0 for ST code, or HDPL1 for all other cases. See Table 119 for more details.  
Reset: system reset  
Register security: no restriction

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

Bits 31:8 Reserved, must be kept at reset value.  
Bits 7:0 **HDPL[7:0]**: temporal isolation level  
This bitfield returns the current temporal isolation level.  
0xB4: HDPL0, RSS  
0x51: HDPL1, iRoT  
0x8A: HDPL2, uRoT  
0x6F: HDPL3, application (secure/non-secure)

### 14.5.3 SBS next HDPL control register (SBS_NEXTHDPLCR)

Address offset: 0x018  
Reset value: 0x0000 0000  
Reset: system reset  
Register security: no restriction

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

Bits 31:2 Reserved, must be kept at reset value.  
Bits 1:0 **NEXTHDPL[1:0]**: index to point to a higher HDPL than the current one  
Index to add to the current HDPL to point (through OBK-HDPL) to the next secure storage areas (OBK-HDPL = HDPL + NEXTHDPL). See Table 120: OBK-HDPL logic for more details.
14.5.4 SBS debug control register (SBS_DBGCR)

Address offset: 0x020
Reset value: 0x0000 0000
Reset: debug reset (system reset or power-on reset)
Register security: HDPL0/1, RAZ/WI otherwise

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

- Bits 31:24 **DBG_AUTH_SEC[7:0]**: control debug opening secure/non-secure
  - Write 0xB4 to this bitfield to open debug for secure and non-secure.
  - Writing any other values only open non-secure.

- Bits 23:16 **DBG_AUTH_HDPL[7:0]**: authenticated debug temporal isolation level
  - Writing to this bitfield defines at which HDPL the authenticated debug opens.
  - 0x51: HDPL1
  - 0x8A: HDPL2
  - 0x6F: HDPL3
  - Note: Writing any other values is ignored. Reading any other value means the debug never opens.

- Bits 15:8 **DBG_UNLOCK[7:0]**: debug unlock when DBG_AUTH_HDPL is reached
  - Write 0xB4 to this bitfield to open the debug when HDPL in SBS_HDPLSR equals to DBG_AUTH_HDPL in this register.

- Bits 7:0 **AP_UNLOCK[7:0]**: access port unlock
  - Write 0xB4 to this bitfield to open the device access port.

14.5.5 SBS debug lock register (SBS_DBGLOCKR)

Address offset: 0x024
Reset value: 0x0000 00B4
Reset: debug reset (system reset or power-on reset)
Register security: HDPL0/1, RAZ/WI otherwise

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

- Bits 31:8 **DBGCFCfg_LOCK[7:0]**: Reserved, must be kept at reset value.
14.5.6 **SBS RSS command register (SBS_RSSCMDR)**

Address offset: 0x034  
Reset value: 0x0000 0000  
Reset: power-on reset  
Register security: always secure (RAZ/WI if non-secure)

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

<table>
<thead>
<tr>
<th>RSSCMD[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **RSSCMD[15:0]: RSS command**  
The application can use this bitfield to pass on a command to the RSS, executed at the next reset.  
When RSSCMD ≠ 0 and PRODUCT_STATE is in Open, then the system always boots on RSS whatever is the boot pin value.

14.5.7 **SBS EPOCH selection control register (SBS_EPOCHSELCR)**

Address offset: 0x0A0  
Reset value: 0x0000 0000  
Reset: system reset  
Register security: Secure when TZ_STATE = 1 (RAZ/WI in non-secure access).  
Non-secure protection when TZ_STATE = 0. This register is protected by privileged whatever is TZ_STATE, RAZ/WI if non privileged access.

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

|----|----|----|----|----|----|----|----|----|----|----|----|----|rw |

Bits 31:2  Reserved, must be kept at reset value.
14.5.8 **SBS security mode configuration control register (SBS_SECCFGR)**

Address offset: 0x0C0

Reset value: 0x0000 0000

Reset: system reset

Register security: always secure. RAZ/WI if non-secure transaction and TZ_STATE = 1. RAZ/WI if TZ_STATE = 0.

This register is programmed by secure software if the user wants functions configurable through system registers to be secure or not.

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

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

Bit 3 **FPUSEC**: FPU security enable
0: SBS_FPUIMP register accessible through secure or non-secure transaction
1: SBS_FPUIMP register accessible only through secure transaction

*Note: This bit can be written only through privilege transaction.*

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CLASSBSEC**: ClassB security enable
0: SBS_CFGR2 register accessible through secure or non-secure transaction
1: SBS_CFGR2 register accessible only through secure transaction

Bit 0 **SBSSEC**: SBS clock control, memory-erase status register and compensation cell register security enable
0: SBS_MESR, SBS_CCCSR, SBS_CCVALR, SBS_CCSWCR registers accessible through secure or non-secure transaction
1: SBS_MESR, SBS_CCCSR, SBS_CCVALR, SBS_CCSWCR registers accessible only through secure transaction
14.5.9 **SBS product mode and configuration register (SBS_PMCR)**

Address offset: 0x100

Reset value: 0x0000 0000

Reset: system reset

Register security:

- When TrustZone is activated (TZEN = 0xB4):
  - depending on ADC1_2_SEC_EN secure input for ANASWVDD and IO_ANA_BOOST_EN configuration bits
  - depending on I/O PBx_SEC_EN secure input for PBx_FMP configuration bits
  - depending on Ethernet ETH_SEC_EN secure input for ETH_SEL_PHY configuration bits
  - depending on SDCE_SEC_EN in SBS_SECCFGR for SMPS_DIV_CLOCK_EN

- When TrustZone is disabled (TZEN = 0xC3), there is no access restriction.

---

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td>ETH_SEL_PHY[2:0]</td>
<td>rw</td>
<td>PB9_FMP</td>
<td>PB8_FMP</td>
<td>PB7_FMP</td>
<td>PB6_FMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td></td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24  Reserved, must be kept at reset value.

Bits 23:21 **ETH_SEL_PHY[2:0]**: Ethernet PHY interface selection

- 000: GMII or MII
- 001: reserved (RGMII)
- 100: RMII
- Other: reserved

Bit 20  Reserved, must be kept at reset value.

Bit 19 **PB9_FMP**: Fast-mode Plus driving capability activation on PB9

This bit can be read and written only with secure access if PB9 is secure in GPIOB. This bit enables the Fm+ driving mode for PB9 when PB9 is not used by I2C peripheral. This can be used to drive a LED for instance.

0: PB9 pin operates in standard mode.
1: Fm+ mode is enabled on PB9 pin and the speed control is bypassed.

Bit 18 **PB8_FMP**: Fast-mode Plus driving capability activation on PB8

This bit can be read and written only with secure access if PB8 is secure in GPIOB. This bit enables the Fm+ driving mode for PB8 when PB8 is not used by I2C peripheral. This can be used to drive a LED for instance.

0: PB8 pin operates in standard mode.
1: Fm+ mode is enabled on PB8 pin and the speed control is bypassed.
Bit 17  **PB7_FMP**: Fast-mode Plus driving capability activation on PB7  
This bit can be read and written only with secure access if PB7 is secure in GPIOB. This bit enables the Fm+ driving mode for PB7 when PB7 is not used by I2C peripheral. This can be used to drive a LED for instance.  
0: PB7 pin operates in standard mode.  
1: Fm+ mode is enabled on PB7 pin and the speed control is bypassed.

Bit 16  **PB6_FMP**: Fast-mode Plus driving capability activation on PB6  
This bit can be read and written only with secure access if PB6 is secure in GPIOB. This bit enables the Fm+ driving mode for PB6 when PB6 is not used by I2C peripheral. This can be used to drive a LED for instance.  
0: PB6 pin operates in standard mode.  
1: Fm+ mode is enabled on PB6 pin and the speed control is bypassed.

Bits 15:8  Reserved, must be kept at reset value.

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

### 14.5.10 SBS FPU interrupt mask register (SBS_FPUIMR)

**Address offset:** 0x104  
**Reset value:** 0x0000 001F  
**Reset:** system reset  
**Register security:** depends on FPUSEC in SBS_SECCFGR  
This register is accessible only through privilege transaction.

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

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

Bits 5:0  **FPU_IE[5:0]**: FPU interrupt enable  
Set and cleared by software to enable the Cortex-M33 FPU interrupts  
FPU_IE[5]: inexact interrupt enable (interrupt disabled at reset)  
FPU_IE[4]: input abnormal interrupt enable  
FPU_IE[3]: overflow interrupt enable  
FPU_IE[2]: underflow interrupt enable  
FPU_IE[1]: divide-by-zero interrupt enable  
FPU_IE[0]: invalid operation interrupt enable
14.5.11 SBS memory erase status register (SBS_MESR)

Address offset: 0x108

Reset value: 0x0000 000X (bit 0 is not affected by system reset)

Register security: depends on SBSSEC in SBS_SECCFGR

<table>
<thead>
<tr>
<th>Bit 31-17</th>
<th>Bit 16 IPMEE</th>
<th>Bit 15-1 Reserved, must be kept at reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>IPMEE</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

Bit 16 **IPMEE**: ICACHE erase status

This bit is set by hardware when ICACHE and PKA RAMs erase is completed after potential tamper detection or a product state regression (refer to Section 47: Tamper and backup registers (TAMP) for more details).

This bit is cleared by software by writing 1 to it.

0: ICACHE and PKA RAM erase on going

1: ICACHE and PKA SRAM erase done

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **MCLR**: device memories erase status

This bit is set by hardware when SRAM2, BKPSRAM, ICACHE, DCACHE and PKA RAMs erase is completed after power-on reset or tamper detection or product state regression (refer to Section 47: Tamper and backup registers (TAMP)).

This bit is not reset by system reset and is cleared by software by writing 1 to it.

0: memory erase on going if not yet cleared by software

1: Memory erase done

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

Address offset: 0x110

Reset value: 0x0000 0000

Reset: system reset

Register security: depends on SBSSEC in SBS_SECCFGR

<table>
<thead>
<tr>
<th>Bit 31-10</th>
<th>Bit 9-0 Reserved, must be kept at reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

Bits 31:10 Reserved, must be kept at reset value.
14.5.13 SBS compensation cell for I/Os value register (SBS_CCVALR)

Address offset: 0x114
Reset value: 0x0000 0088
Reset: system reset
Register security: depends on SBSSEC in SBS_SECCFG

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 APSRC2[3:0]: compensation value for the PMOS transistor
This value is provided by the cell and must be interpreted by the processor to compensate the slew rate in the functional range.

Note: The table provided in the document seems to be incorrect or incomplete. The correct interpretation of the bits should be based on the descriptions provided in the text.
Bits 11:8 \textbf{ANSRC2[3:0]}: Compensation value for the NMOS transistor
This value is provided by the cell and must be interpreted by the processor to compensate the slew rate in the functional range.

Bits 7:4 \textbf{APSRC1[3:0]}: compensation value for the PMOS transistor
This value is provided by the cell and must be interpreted by the processor to compensate the slew rate in the functional range.

Bits 3:0 \textbf{ANSRC1[3:0]}: compensation value for the NMOS transistor
This value is provided by the cell and must be interpreted by the processor to compensate the slew rate in the functional range.

14.5.14 \textbf{SBS compensation cell for I/Os software code register (SBS_CCSWCR)}

Address offset: 0x118
Reset value: 0x0000 7878
Reset: system reset
Register security: depends on SBSSEC in SBS_SECCFGR

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 \textbf{SW_APSRC2[3:0]}: PMOS compensation code for the VDDIO power rails
This bitfield is written by software to define an I/O compensation cell code for PMOS transistors of the VDDIO power rail. This code is applied to the I/O when CS2 is set in SBS_CCSR.

Bits 11:8 \textbf{SW_ANSRC2[3:0]}: NMOS compensation code for VDDIO power rails
This bitfield is written by software to define an I/O compensation cell code for NMOS transistors of the VDD power rail. This code is applied to the I/O when CS2 is set in SBS_CCSR.

Bits 7:4 \textbf{SW_APSRC1[3:0]}: PMOS compensation code for the VDD power rails
This bitfield is written by software to define an I/O compensation cell code for PMOS transistors of the VDDIO power rail. This code is applied to the I/O when CS1 is set in SBS_CCSR.

Bits 3:0 \textbf{SW_ANSRC1[3:0]}: NMOS compensation code for VDD power rails
This bitfield is written by software to define an I/O compensation cell code for NMOS transistors of the VDD power rail. This code is applied to the I/O when CS1 is set in SBS_CCSR.
### 14.5.15 SBS Class B register (SBS_CFGR2)

Address offset: 0x120  
Reset value: 0x0000 0000  
Reset: system reset

Register security: depends on CLASSBSEC in SBS_SECCFGR

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

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

**Bit 3 ECCL**: ECC lock  
This bit is set and cleared by software. It can be used to enable and lock the flash memory double ECC error with break input of TIM1/8/15/6/17.  
0: double ECC error flag disconnected to timer break inputs  
1: double ECC error flag connected to timer break inputs

**Bit 2 PVDL**: PVD lock  
This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection with TIM1/8/15/16/17 break inputs.  
0: PVD interrupt disconnected from timer break inputs. PVD_EN and PVD_SEL[2:0] in the PWR registers are read/write.  
1: PVD interrupt is connected to timer break inputs. PVD_EN and PVD_SEL[2:0] in the PWR registers are read only

**Bit 1 SEL**: SRAM ECC error lock  
This bit is set by software and cleared only by a system reset. It can be used to enable and lock the SRAM double ECC error signal with break input of TIM1/8/15/16/17.  
0: SRAM double ECC error flag disconnected from timer break inputs  
1: SRAM double ECC error flag connected to timer break inputs

**Bit 0 CLL**: core lockup lock  
This bit is set by software and cleared only by a system reset. It can be used to enable and lock the lockup (HardFault) output of Cortex-M33 with TIM1/8/15/16/17 break inputs.  
0: lockup output disconnected from timer break inputs  
1: lockup output connected to timer break inputs
14.5.16  SBS CPU non-secure lock register (SBS_CNSLCKR)

Address offset: 0x144
Reset value: 0x0000 0000
Reset: system reset

Register security: This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

This register is used to lock the configuration of non-secure MPU and VTOR_NS registers of the Cortex-M33.

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

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

Bit 1  LOCKNSMPU: non-secure MPU register lock
This bit is set by software and cleared only by a system reset. When set, this bit disables write access to non-secure MPU_CTRL_NS, MPU_RNR_NS and MPU_RBAR_NS registers.
0: non-secure MPU registers write enabled
1: non-secure MPU registers write disabled

Bit 0  LOCKNSVTOR: VTOR_NS register lock
This bit is set by software and cleared only by a system reset.
0: VTOR_NS register write enabled
1: VTOR_NS register write disabled
14.5.17 SBS CPU secure lock register (SBS_CSLCKR)

Address offset: 0x148
Reset value: 0x0000 0000

Reset: system reset

Register security: This register can be written only when the access is secure/privilege.

A non-secure read/write access is RAZ/WI and generates an illegal access event. When the system is not secure (TZ_STATE = 0), this register is RAZ/WI.

This register is used to lock the configuration of PRIS and BFHFNMINS in the AIRCR, SAU, secure MPU and VTOR_S registers of the Cortex-M33.

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

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

Bit 2 LOCKSAU: SAU registers lock
This bit is set by software and cleared only by a system reset. When set, this bit disables write access to SAU_CTRL, SAU_RNR, SAU_RBAR and SAU_RLAR registers.
0: SAU registers write enabled
1: SAU registers write disabled

Bit 1 LOCKSMPU: secure MPU registers lock
This bit is set by software and cleared only by a system reset. When set, this bit disables write access to secure MPU_CTRL, MPU_RNR and MPU_RBAR registers.
0: Secure MPU registers writes enabled
1: Secure MPU registers writes disabled

Bit 0 LOCKSVTAIRCR: VTOR_S and AIRCR register lock
This bit is set by software and cleared only by a system reset. When set, this bit disables write access to VTOR_S register, PRIS and BFHFNMINS bits in the AIRCR register.
0: VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write enabled
1: VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write disabled
14.5.18  SBS flit ECC NMI mask register (SBS_ECCNMIr)

Address offset: 0x14C
Reset value: 0x0000 0000
Reset: system reset

Register security: secure access only when TZ_STATE = 1 (RAZ/WI in non-secure access).
No security protection if TZ_STATE = 0.

This register is accessible only through privilege transaction.

This register sets up the expected behavior on NMI regarding double ECC errors from the flash memory.

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

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

Bit 0  **ECCNMI_MASK_EN**: NMI behavior setup when a double ECC error occurs on flitf data part
0: NMI generated if a double ECC error in the flitf data part
1: NMI not generated if a double ECC error in the flitf data part
### 14.5.19 SBS register map

#### Table 121. SBS register map and reset values

| Offset | Register               | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x010  | SBS_HDPLCR             | INCR_HDPL[7:0] | 1  | 0  | 1  | 1  | 0  | 1  | 0  | 0  |
| 0x014  | SBS_HDPLSR             | HDPL[7:0]      |     |    |    |    |    |    |    |    |
| 0x018  | SBS_NEXTHDPLCR         | Reserved       |     |    |    |    |    |    |    |    |
| 0x020  | SBS_DBGCR              | DBG_AUTH_SEC[7:0] | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x024  | SBS_DBGLOCKCR          | DBGCFG_LOCK[7:0] |     |    |    |    |    |    |    |    |
| 0x028 to 0x030 | Reserved | Reserved |    |    |    |    |    |    |    |    |
| 0x034  | SBS_RSSCMDR            | RSSCMD[15:0]   |     |    |    |    |    |    |    |    |
| 0x038 to 0x09C | Reserved | Reserved |    |    |    |    |    |    |    |    |
| 0x0A0  | SBS_EPOCHSELCR         | Reserved       |     |    |    |    |    |    |    |    |
| 0x0A4 to 0x0BC | Reserved | Reserved |    |    |    |    |    |    |    |    |
| 0x0CC  | SBS_SECCFGR            | Reserved       |     |    |    |    |    |    |    |    |
| 0x0C4 to 0x0FC | Reserved | Reserved |    |    |    |    |    |    |    |    |
| 0x100  | SBS_PMCR               | ETH_SEL_PHY[2:0] | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x104  | SBS_FPUIMR             | FPU_JE[5:0]    |     |    |    |    |    |    |    |    |
| 0x108  | SBS_MESR               | IP_MEE         |     |    |    |    |    |    |    |    |
| 0x110  | SBS_CCCSR              | Reserved       |     |    |    |    |    |    |    |    |
Refer to Section 2.3 for the register boundary addresses.
15 Peripherals interconnect matrix

15.1 Interconnect matrix introduction

Several peripherals have direct connections between them, enabling autonomous communication and/or synchronization: this approach saves CPU resources, and power supply consumption. In addition, these hardware connections remove software latency and help the design of predictable system.

Depending on peripherals, these interconnections can operate in Run, Sleep, and Stop modes.
## 15.2 Connection summary

Table 122. Peripherals interconnect matrix\(^{(1)}\) \(^{(2)}\)

<table>
<thead>
<tr>
<th>Source</th>
<th>TIM1</th>
<th>TIM2</th>
<th>TIM3</th>
<th>TIM4</th>
<th>TIM5</th>
<th>TIM6</th>
<th>TIM7</th>
<th>TIM8</th>
<th>TIM12</th>
<th>TIM13</th>
<th>TIM14</th>
<th>TIM15</th>
<th>TIM16</th>
<th>TIM17</th>
<th>LPTIM1/2</th>
<th>LPTIM3/4/5/6</th>
<th>ADC1</th>
<th>ADC2</th>
<th>GPDMA1/2</th>
<th>EXTI</th>
<th>RTC wake-up</th>
<th>RTC alarm</th>
<th>TAMP</th>
<th>HSE</th>
<th>LSE</th>
<th>CSS in LSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM1</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2 4</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>3 3</td>
<td>3 3</td>
<td>-</td>
<td>-</td>
<td>6 6 6</td>
<td>6 6 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM2</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2 4</td>
<td>2 4</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>3 3</td>
<td>3 3</td>
<td>-</td>
<td>-</td>
<td>6 6 6</td>
<td>6 6 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM3</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6 6 6</td>
<td>6 6 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM4</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>13</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM5</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2 4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM6</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM7</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM8</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM9</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM10</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM11</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM12</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM13</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM14</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2 4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM15</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM16</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM17</td>
<td>- 1</td>
<td>1 1</td>
<td>1 1</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>1 1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPTIM1/2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2 4</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPTIM3/4/5/6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC1</td>
<td>3 3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC2</td>
<td>3 3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPDMA1/2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXTI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6 6 6</td>
<td>2 4 10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTC wake-up</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTC alarm</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6 6 6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6 6 6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10 14 15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSE</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>12</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSS in LSE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>13</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 122. Peripherals interconnect matrix(1) (2) (continued)

<table>
<thead>
<tr>
<th>Source</th>
<th>TIM1</th>
<th>TIM8</th>
<th>TIM2</th>
<th>TIM3</th>
<th>TIM4</th>
<th>TIM5</th>
<th>TIM6</th>
<th>TIM7</th>
<th>TIM12</th>
<th>TIM15</th>
<th>TIM16</th>
<th>TIM17</th>
<th>LPTIM1/2/3</th>
<th>LPTIM4</th>
<th>LPTIM5/6</th>
<th>ADC1/2</th>
<th>DAC1/2</th>
<th>GPDMA1/2</th>
<th>TAM</th>
<th>RTC</th>
<th>AES/SAES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HSI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LSI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MCO1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MCO2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V(&quot;CORE&quot;)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V(&quot;REFINT&quot;)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V(&quot;sensor&quot;)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V(&quot;BAT&quot;)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V(&quot;BAT&quot;) / temperature monitor</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>System errors</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>System flash memory</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AES/SAES</td>
<td>-</td>
<td>-</td>
<td>9</td>
<td>9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1. Numbers in this table are links to corresponding subsections of Section 15.3.
2. "-" means no interconnect.
15.3  Interconnection details

15.3.1  Master to slave interconnection for timers

From timer (TIM1/2/3/4/5/8/12/13/14/15/16/17) to timer (TIM1/2/3/4/5/8/12/15).

Purpose

Some timers are linked together internally for synchronization or chaining.

When one timer is configured in master mode, it can reset, start, stop, or clock the counter of another timer configured in slave mode.

A description of the feature is provided in Section 39.4.23: Timer synchronization.

The synchronization modes are detailed in:

- Section 38.3.30 for advanced-control timers TIM1/8
- Section 39.4.22 for general-purpose timers TIM2/3/4/5
- Section 41 for the general-purpose timers TIM12/13/14
- Section 42.4.23 for the general-purpose timer TIM15

Triggering signals

The output (from master) is on signal TIMx_TRGO (and TIMx_TRGO2 for TIM1/8) following a configurable timer event. It can be also from signals tim16_oc1 and tim17_oc1 in case of TIM16/17. The input (to slave) is on signals TIMx_ITR0/1/2/3.

The possible master/slave connections are given in:

- Table 381 for advanced-control timers TIM1/8
- Table 404 for general-purpose timers TIM2/3/4/5
- Table 425 for the general-purpose timers TIM12/13/14
- Table 437 for the general-purpose timer TIM15

Active power mode

Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:

- Table 393: Effect of low-power modes on TIM1/TIM8
- Table 411: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5
- Table 427: Effect of low-power modes on TIM12/TIM13/TIM14
- Table 443: Effect of low-power modes on TIM15/TIM16/TIM17

15.3.2  Triggers to ADCs

From EXTI, timers (TIM1/2/3/4/6/8/15) and LP timers (LPTIM1/2) to ADC1/ADC2.

Purpose

A conversion (or a sequence of conversions) can be triggered either by software or by an external event (such as timer capture or input pins). For ADC12, if the EXTEN[1:0] (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, external events can trigger a conversion with the selected polarity.
More details in:
- **Section 26.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)**
- EXTEN[1:0] defined in *ADC configuration register (ADC_CFGR)*
- JEXTEN[1:0] defined in *ADC injected sequence register (ADC_JSQR)*

General-purpose timers (TIM2/3/4), basic timer (TIM6), advanced-control timers (TIM1/8), and general-purpose timer (TIM15) can be used to generate the ADC triggering event through the timer outputs tim_oc and tim_trgo.

Low-power timers (LPTIM1/2) can be used to generate the ADC triggering event through the LPTIM channels, in addition to the EXTI on channels 11 and 15.

### Triggering signals

For ADC1/ADC2, the input triggering signals and the description of the interconnection between ADC1/ADC2 and timers are given in:

- adc_ext_trg: **Table 242: ADC interconnection**
- adc_jext_trg: **Table 242: ADC interconnection**
- **Section 26.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)**
- **Section 26.4.25: Timing diagrams example (single/continuous modes, hardware/software triggers)**

### Active power mode

This interconnection is active in Run and Sleep modes for all ADCs. The timers are active only in Run and Sleep modes. The effects of low-power modes are given in:

- **Table 393: Effect of low-power modes on TIM1/TIM8**
- **Table 411: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5**
- **Table 443: Effect of low-power modes on TIM15/TIM16/TIM17**
- **Table 449: STM32H563/H573 and STM32H562 LPTIM features**
- **Table 464: Effect of low-power modes on the LPTIM**

### 15.3.3 ADC analog watchdogs as triggers to timers

From ADC1/ADC2 to TIM1/8.

**Purpose**

The internal analog watchdog output signals coming from ADC1/ADC2 are connected to on-chip timers. ADC1/ADC2 can provide trigger events through analog watchdog signals to advanced-control timers (TIM1/8) to reset, start, stop, or clock the counter.

Settings description of the ADC analog watchdog and timer trigger, are provided in:

- **Section 38.3.6: External trigger input for TIM1/8**
- **Table 382** for the internal ADC1/ADC2 sources connected to TIM1/8 (tim_etrm) input multiplexer
- **Section 26.4.28** for the ADC1/ADC2/ADC_AWDy_OUT signal output generation
Triggering signals
The output (from ADC) is on signals ADCn_AWDx_OUT, with n being the ADC instance and x = 1, 2, 3 (three watchdogs per ADC). The input (to timer) is on signal TIMx_ETR (external trigger).

Active power mode
ADC1/ADC2 are active in Run and Sleep modes.

15.3.4 Triggers to DAC
From timer (TIM1/2/4/5/6/7/8/15), low-power timers (LPTIM1/2), and EXTI to DAC.

Purpose
General-purpose timers (TIM2/4/5/15), basic timers (TIM6/7), advanced-control timers (TIM1/8), LP timers (LPTIM1/2) outputs channels (lptim1_ch1 and lptim2_ch1), and EXTI can be used as triggering event to start a DAC conversion.

Triggering signals
The output (from timer) on the TIMx_TRGO signal and from LP timers are directly connected to corresponding DAC inputs.

The selection of input triggers on DAC is provided in:
- Table 266: DAC interconnection
- Section 28.4.8: DAC trigger selection

Active power mode
This interconnect is active in Run, Sleep, and Stop modes.

15.3.5 Clock sources to timers
From HSE, LSE, LSI, HSI, and MCO to timers (TIM2/12/15/16/17) and LP timers (LPTIM1/2).

Purpose
A timer input or counter can receive different clock sources, and can be used, for example, to calibrate the internal oscillator on a reference clock.

External clocks (HSE, LSE), internal clocks (LSI, CSI, HSI), and microcontroller output clock (MCO) can be used as input to timers:
- LSE, HSI, and CSI are assigned to general purpose timer TIM2 as external inputs signals. LSE can be selected as counter clock provided by an external clock source in mode1 (tim_ti1_in) and mode2 (external trigger input tim_etr_in). Inputs assignment and clock selection description are detailed in:
  - Section 39.4.5: Clock selection for TIM2
  - External clock mode1: Table 423: Interconnect to the tim_ti1 input multiplexer for TIM12, tim_ti1_in4 (HSI), and tim_ti1_in5 (CSI)
  - External clock mode2: Table 405: Interconnect to the tim_etr input multiplexer for tim_etr3 (LSE)
• LSE, LSI, CSI, and HSE are assigned to general purpose timers TIM15/16/17 as external inputs signals. LSE/LSI/CSI/HSE can be selected as counter clock provided by an external clock source in mode1 (tim_t1 or tim_t2 signals). Inputs assignment and clock selection description are detailed in:
  – *Section 42.4.6: Clock selection* for TIM15/16/17. External clock mode1: external input pin (tim_t1 or tim_t2, if available)
  – *Table 423: Interconnect to the tim_t1 input multiplexer*, tim_t1_in1 (LSI-TIM16), tim_t1_in2 (LSE-TIM16/HSE-TIM17), tim_t1_in4 (LSE-TIM15), and tim_t1_in5 (CSI-TIM15)

• Microcontroller output clock (MCO): MCO1 is connected as external input to general-purpose timer TIM17, MCO2 is connected as external input to general-purpose timer TIM15.
  – *Table 435: Interconnect to the tim_t1 input multiplexer* for TIM15/TIM16/TIM17

• LSI and LSE can be selected as input capture 2 to LPTIM1 as described in *Table 458: LPTIM1 input capture 2 connection*.

• HSI/1024, CSI/128, and HSI/8 can be selected as input capture 2 to LPTIM2 as described in *Table 459: LPTIM2 input capture 2 connection*.

**Triggering signals**

Lptim_ic2_mux1 LPTIM input capture selection can be set in the LPTIM configuration register 2 (LPTIM_CFGR2). For timers, the internal clock signal can be selected as counter clock provided by an external clock source in mode1 (tim_t1_in) and mode2 (external trigger input tim_etr_in).

**Active power mode**

This feature is available under Run and Sleep modes.

### 15.3.6 Triggers to low-power timers

From EXTI, TAMP, and RTC alarm to LP timers (LPTIM1/2/3/4/5/6).

**Purpose**

LPTIM1/2/3/4/5/6 counters can be started either by software, or after the detection of an active edge on one of the eight trigger inputs (see *Section 43.4.7: Trigger multiplexer*).

GPIO can also be selected as LPTIM input capture selection or LPTIM input selection, according to the LPTIM configuration register 2 (LPTIM_CFGR2).

**Triggering signals**

This trigger feature is described in *Section 43.4.7: Trigger multiplexer* and the following sections. The input selection is described in *Table 454: LPTIM1/2/3/4/5/6 external trigger connection*.

**Active power mode**

This interconnection is active in Run, Sleep, and Stop modes.

### 15.3.7 RTC wake-up as inputs to timers

From RTC to timer (TIM16).
Purpose
RTC wake-up interrupt can be used as input to general-purpose timer (TIM16) channel 1.

Triggering signals
RTC wake-up signal is connected to tim_ti1_in3 signal as described in Table 435: Interconnect to the tim_ti1 input multiplexer for TIM16.

Active power mode
This interconnection is active down to Stop mode. Timers are not active but the count is performed at wake-up.

15.3.8 System errors as break signals to timers
From system errors to timers (TIM1/8/15/16/17).

Purpose
CSS, CPU lockup, SRAM2/3 ECC double errors, SRAM1 parity errors, FLASH ECC double-error detection, and PVD can generate system errors in the form of timer break toward timers (TIM1/8/15/16/17).

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers.

Triggering signals
The possible sources of break are described in:
- Section 38.3.18: Using the break function for TIM1/8
- Section 42.4.15: Using the break function for TIM15/16/17
- Table 385: System break interconnect for TIM1/8
- Table 439: System break interconnect for TIM15/16/17

Active power mode
Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:
- Table 393: Effect of low-power modes on TIM1/TIM8
- Table 411: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5
- Table 443: Effect of low-power modes on TIM15/TIM16/TIM17

15.3.9 Triggers for communication peripherals
From Ethernet to timers (TIM2/TIM3).

Purpose
To synchronize system clock with network, internal connections are available between timers and PTP to check for clock drifts.
Triggering signals

- The outputs (from timer) are directly connected to Ethernet PTP triggers inputs. More details are given in the sections below:
  - Section 57.3: Ethernet pins and internal signals
  - Table 400: Interconnect to the tim_t1 input multiplexer
  - Table 405: Interconnect to the tim_et input multiplexer

Active power mode

These interconnections remain active in Run and Sleep modes.

15.3.10 Triggers to GPDMA1/2

From EXTI, RTC (alarm/wake-up), TAMP, timers (TIM2/12/15), LP timers (LPTIM1/2/3/4/5/6), GPDMA1 transfer complete (gpdma1_chx_tc, gpdma2_chx_tcf) to GPDMA1/2.

Purpose

A GPDMA trigger can be assigned to GPDMA channel x. A programmed GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event. The trigger mode can also be programmed to condition the LLI link transfer. More details are given in the sections below:
  - Section 16.3.7: GPDMA triggers
  - Section 16.4.12: GPDMA triggered transfer
  - GPDMA channel x transfer register 2 (GPDMA_CxTR2) for more details on:
    - Trigger selection TRIGSEL[5:0] field
    - Trigger mode (LLI) defined by TRIGM[1:0]
    - Trigger polarity as defined by TRIGPOL[1:0]

Triggering signals

GPDMA trigger mapping is specified in Table 129: Programmed GPDMA1/2 trigger, according to GPDMA_CxTR2.TRIGSEL[5:0].

Active power mode

This interconnection remains functional in Sleep mode.

Refer to:
  - Section 16.6: GPDMA in low-power modes

15.3.11 Internal analog signals to analog peripherals

From internal analog source to ADC (ADC1/2).

Purpose

The internal reference voltage \(V_{\text{REFINT}}\), the internal temperature sensor \(V_{\text{SENSE}}\), the internal digital core voltage \(V_{\text{DDCORE}}\), and the \(V_{\text{BAT}}\) monitoring channel are connected to ADC (ADC1/2) input channels.
This is according to:

- Section 26.2: ADC main features
- Section 26.4.11: Channel selection (SQRx, JSQRx)

15.3.12 Clock source for the DAC sample and hold mode

From LSI/LSE to DAC1.

**Purpose**

DAC1 can run in Stop mode. The sample and hold block and its associated registers use the LSI or LSE clock source (dac_hold_ck) in Stop mode.

*Table 265: DAC internal input/output signals:* dac_hold_ck, Input, DAC low-power clock used in sample and hold mode

**Active power mode**

This feature remains available in Run, Sleep and Stop modes.

15.3.13 Internal tamper sources

From internal peripherals, clocks, or monitoring, to tamper.

**Purpose**

To detect any abnormal activity or tentative to corrupt the device, embedded tampers alert the system of undesired events. Different actions can be taken as consequence.

The list of tamper sources can be found in *Table 489: TAMP interconnection.*

**Active power mode**

This interconnection is active in all power modes if the tamper source is activated.

15.3.14 Output from tamper to RTC

From TAMP to RTC.

**Purpose**

The RTC can timestamp a tamper event to retrieve history of the detection. The RTC can also control GPIOs, and set a signal based on tamper or alarm status outside the MCU.

Refer to section *Section 46.3.3: GPIOs controlled by the RTC and TAMP* for more details.

**Active power mode**

This interconnection remains active in all power modes.
15.3.15 Encryption keys to AES/SAES

From TAMP backup registers, system flash memory to and in between SAES and AES.

Purpose

The encryption mechanism requires an hardware key that must be stored in a protected non-volatile memory. Different approaches are implemented to load them in a non-readable way. Tamper backup registers or system flash memory can be used to store respectively BHK or RHUK, and to implement a dedicated bus to pass it to the SAES.

Refer to Section 34.4.14: SAES operation with wrapped keys for more details.

The AES encryption mechanism (faster than the SAES) can benefit from the sharing key of the SAES. Refer to Section 34.4.15: SAES operation with shared keys for more details.

Active power mode

AES and SAES are operating under Run and Sleep modes.
16 General purpose direct memory access controller (GPDMA)

16.1 GPDMA introduction

The general purpose direct memory access (GPDMA) controller is a bus master and system peripheral.

The GPDMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories via linked-lists, upon the control of an off-loaded CPU.

16.2 GPDMA main features

- Dual bidirectional AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral
- Autonomous data transfers during low-power modes (see Section 16.3.3)
  Transfer arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half transfer complete, data transfer error, user setting error, link transfer error, completed suspension, and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 8 concurrent GPDMA channels:
  - Per channel FIFO for queuing source and destination transfers (see Section 16.3.2)
  - Intra-channel GPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel GPDMA transfers chaining via programmable GPDMA input triggers connection to GPDMA task completion events
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based reordering, packing or unpacking, padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level
- Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at a block level, between successive burst transfers
- 2D source and destination addressing: programmable signed address offsets between successive burst transfers (non-contiguous addressing within a block, combined with programmable signed address offsets between successive blocks, at a second 2D/repeated block level, for a reduced set of channels (see Section 16.3.2)
- Support for scatter-gather (multi-buffer transfers), data interleaving and deinterleaving via 2D addressing
- Programmable GPDMA request and trigger selection
- Programmable GPDMA half transfer and transfer complete events generation
- Pointer to the next linked-list item and its data structure in memory, with automatic update of the GPDMA linked-list control registers

- **Debug:**
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags

- **TrustZone support:**
  - Support for secure and nonsecure GPDMA transfers, independently at a first channel level, and independently at a source/destination and link sublevels
  - Secure and nonsecure interrupts reporting, resulting from any of the respectively secure and nonsecure channels
  - TrustZone-aware AHB slave port, protecting any GPDMA secure resource (register, register field) from a nonsecure access

- **Privileged/unprivileged support:**
  - Support for privileged and unprivileged GPDMA transfers, independently at a channel level
  - Privileged-aware AHB slave port

### 16.3 GPDMA implementation

#### 16.3.1 GPDMA instances

There are two instances of the GPDMA in the devices, named as GPDMA1 and GPDMA2. Each GPDMA instance has the same channel-based implementation and is connected to the same requests and triggers, as detailed in the following sub-sections.

#### 16.3.2 GPDMA channels

A given GPDMA channel $x$ is implemented with the following features and intended usage. To make the best use of the GPDMA performances, the table below lists some general recommendations, allowing the user to select and allocate a channel, given its implemented FIFO size and the requested GPDMA transfer.
### 16.3.3 GPDMA autonomous mode in low-power modes

The GPDMA autonomous mode and wake-up feature are implemented in the device low-power modes as per the table below.

**Table 124. GPDMA1/2 autonomous mode and wake-up in low-power modes**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Low-power modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonomous mode and wake-up</td>
<td>GPDMA1/2 in Sleep mode</td>
</tr>
</tbody>
</table>

### 16.3.4 GPDMA requests

A GPDMA request from a peripheral can be assigned to a GPDMA channel x, via REQSEL[7:0] in GPDMA_CxTR2, provided that SWREQ = 0.

The GPDMA requests mapping is specified in the table below.

**Table 125. Programmed GPDMA1/2 request**

<table>
<thead>
<tr>
<th>GPDMA_CxTR2.REQSEL[7:0]</th>
<th>Selected GPDMA request</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>adc1_dma</td>
</tr>
<tr>
<td>1</td>
<td>adc2_dma</td>
</tr>
<tr>
<td>2</td>
<td>dac1_ch1_dma</td>
</tr>
<tr>
<td>3</td>
<td>dac1_ch2_dma</td>
</tr>
<tr>
<td>GPDMA_CxTR2.REQSEL[7:0]</td>
<td>Selected GPDMA request</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>4</td>
<td>tim6_upd_dma</td>
</tr>
<tr>
<td>5</td>
<td>tim7_upd_dma</td>
</tr>
<tr>
<td>6</td>
<td>spi1_rx_dma</td>
</tr>
<tr>
<td>7</td>
<td>spi1_tx_dma</td>
</tr>
<tr>
<td>8</td>
<td>spi2_rx_dma</td>
</tr>
<tr>
<td>9</td>
<td>spi2_tx_dma</td>
</tr>
<tr>
<td>10</td>
<td>spi3_rx_dma</td>
</tr>
<tr>
<td>11</td>
<td>spi3_tx_dma</td>
</tr>
<tr>
<td>12</td>
<td>i2c1_rx_dma</td>
</tr>
<tr>
<td>13</td>
<td>i2c1_tx_dma</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>i2c2_rx_dma</td>
</tr>
<tr>
<td>16</td>
<td>i2c2_tx_dma</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
</tr>
<tr>
<td>18</td>
<td>i2c3_rx_dma</td>
</tr>
<tr>
<td>19</td>
<td>i2c3_tx_dma</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
</tr>
<tr>
<td>21</td>
<td>usart1_rx_dma</td>
</tr>
<tr>
<td>22</td>
<td>usart1_tx_dma</td>
</tr>
<tr>
<td>23</td>
<td>usart2_rx_dma</td>
</tr>
<tr>
<td>24</td>
<td>usart2_tx_dma</td>
</tr>
<tr>
<td>25</td>
<td>usart3_rx_dma</td>
</tr>
<tr>
<td>26</td>
<td>usart3_tx_dma</td>
</tr>
<tr>
<td>27</td>
<td>uart4_rx_dma</td>
</tr>
<tr>
<td>28</td>
<td>uart4_tx_dma</td>
</tr>
<tr>
<td>29</td>
<td>uart5_rx_dma</td>
</tr>
<tr>
<td>30</td>
<td>uart5_tx_dma</td>
</tr>
<tr>
<td>31</td>
<td>usart6_rx_dma</td>
</tr>
<tr>
<td>32</td>
<td>usart6_tx_dma</td>
</tr>
<tr>
<td>33</td>
<td>uart7_rx_dma</td>
</tr>
<tr>
<td>34</td>
<td>uart7_tx_dma</td>
</tr>
<tr>
<td>35</td>
<td>uart8_rx_dma</td>
</tr>
<tr>
<td>36</td>
<td>uart8_tx_dma</td>
</tr>
<tr>
<td>37</td>
<td>uart9_rx_dma</td>
</tr>
<tr>
<td>38</td>
<td>uart9_tx_dma</td>
</tr>
<tr>
<td>GPDMA_CxTR2.REQSEL[7:0]</td>
<td>Selected GPDMA request</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>39</td>
<td>uart10_rx_dma</td>
</tr>
<tr>
<td>40</td>
<td>uart10_tx_dma</td>
</tr>
<tr>
<td>41</td>
<td>uart11_rx_dma</td>
</tr>
<tr>
<td>42</td>
<td>uart11_tx_dma</td>
</tr>
<tr>
<td>43</td>
<td>uart12_rx_dma</td>
</tr>
<tr>
<td>44</td>
<td>uart12_tx_dma</td>
</tr>
<tr>
<td>45</td>
<td>lpuart1_rx_dma</td>
</tr>
<tr>
<td>46</td>
<td>lpuart1_tx_dma</td>
</tr>
<tr>
<td>47</td>
<td>spi4_rx_dma</td>
</tr>
<tr>
<td>48</td>
<td>spi4_tx_dma</td>
</tr>
<tr>
<td>49</td>
<td>spi5_rx_dma</td>
</tr>
<tr>
<td>50</td>
<td>spi5_tx_dma</td>
</tr>
<tr>
<td>51</td>
<td>spi6_rx_dma</td>
</tr>
<tr>
<td>52</td>
<td>spi6_tx_dma</td>
</tr>
<tr>
<td>53</td>
<td>sai1_a_dma</td>
</tr>
<tr>
<td>54</td>
<td>sai1_b_dma</td>
</tr>
<tr>
<td>55</td>
<td>sai2_a_dma</td>
</tr>
<tr>
<td>56</td>
<td>sai2_b_dma</td>
</tr>
<tr>
<td>57</td>
<td>ospi1_dma</td>
</tr>
<tr>
<td>58</td>
<td>tim1_cc1_dma</td>
</tr>
<tr>
<td>59</td>
<td>tim1_cc2_dma</td>
</tr>
<tr>
<td>60</td>
<td>tim1_cc3_dma</td>
</tr>
<tr>
<td>61</td>
<td>tim1_cc4_dma</td>
</tr>
<tr>
<td>62</td>
<td>tim1_upd_dma</td>
</tr>
<tr>
<td>63</td>
<td>tim1_trg_dma</td>
</tr>
<tr>
<td>64</td>
<td>tim1_com_dma</td>
</tr>
<tr>
<td>65</td>
<td>tim8_cc1_dma</td>
</tr>
<tr>
<td>66</td>
<td>tim8_cc2_dma</td>
</tr>
<tr>
<td>67</td>
<td>tim8_cc3_dma</td>
</tr>
<tr>
<td>68</td>
<td>tim8_cc4_dma</td>
</tr>
<tr>
<td>69</td>
<td>tim8_upd_dma</td>
</tr>
<tr>
<td>70</td>
<td>tim8_tig_dma</td>
</tr>
<tr>
<td>71</td>
<td>tim8_com_dma</td>
</tr>
<tr>
<td>72</td>
<td>tim2_cc1_dma</td>
</tr>
<tr>
<td>73</td>
<td>tim2_cc2_dma</td>
</tr>
</tbody>
</table>
Table 125. Programmed GPDMA1/2 request (continued)

<table>
<thead>
<tr>
<th>GPDMA_CxTR2.REQSEL[7:0]</th>
<th>Selected GPDMA request</th>
</tr>
</thead>
<tbody>
<tr>
<td>74</td>
<td>tim2_cc3_dma</td>
</tr>
<tr>
<td>75</td>
<td>tim2_cc4_dma</td>
</tr>
<tr>
<td>76</td>
<td>tim2_upd_dma</td>
</tr>
<tr>
<td>77</td>
<td>tim3_cc1_dma</td>
</tr>
<tr>
<td>78</td>
<td>tim3_cc2_dma</td>
</tr>
<tr>
<td>79</td>
<td>tim3_cc3_dma</td>
</tr>
<tr>
<td>80</td>
<td>tim3_cc4_dma</td>
</tr>
<tr>
<td>81</td>
<td>tim3_upd_dma</td>
</tr>
<tr>
<td>82</td>
<td>tim3_trg_dma</td>
</tr>
<tr>
<td>83</td>
<td>tim4_cc1_dma</td>
</tr>
<tr>
<td>84</td>
<td>tim4_cc2_dma</td>
</tr>
<tr>
<td>85</td>
<td>tim4_cc3_dma</td>
</tr>
<tr>
<td>86</td>
<td>tim4_cc4_dma</td>
</tr>
<tr>
<td>87</td>
<td>tim4_upd_dma</td>
</tr>
<tr>
<td>88</td>
<td>tim5_cc1_dma</td>
</tr>
<tr>
<td>89</td>
<td>tim5_cc2_dma</td>
</tr>
<tr>
<td>90</td>
<td>tim5_cc3_dma</td>
</tr>
<tr>
<td>91</td>
<td>tim5_cc4_dma</td>
</tr>
<tr>
<td>92</td>
<td>tim5_upd_dma</td>
</tr>
<tr>
<td>93</td>
<td>tim5_trg_dma</td>
</tr>
<tr>
<td>94</td>
<td>tim15_cc1_dma</td>
</tr>
<tr>
<td>95</td>
<td>tim15_upd_dma</td>
</tr>
<tr>
<td>96</td>
<td>tim15_trg_dma</td>
</tr>
<tr>
<td>97</td>
<td>tim15_com_dma</td>
</tr>
<tr>
<td>98</td>
<td>tim16_cc1_dma</td>
</tr>
<tr>
<td>99</td>
<td>tim16_upd_dma</td>
</tr>
<tr>
<td>100</td>
<td>tim17_cc1_dma</td>
</tr>
<tr>
<td>101</td>
<td>tim17_upd_dma</td>
</tr>
<tr>
<td>102</td>
<td>lptim1_ic1_dma</td>
</tr>
<tr>
<td>103</td>
<td>lptim1_ic2_dma</td>
</tr>
<tr>
<td>104</td>
<td>lptim1_ue_dma</td>
</tr>
<tr>
<td>105</td>
<td>lptim2_ic1_dma</td>
</tr>
<tr>
<td>106</td>
<td>lptim2_ic2_dma</td>
</tr>
<tr>
<td>107</td>
<td>lptim2_ue_dma</td>
</tr>
<tr>
<td>108</td>
<td>dcmi_dma or pssi_dma(1)</td>
</tr>
</tbody>
</table>
16.3.5  GPDMA block requests

Some GPDMA requests must be programmed as a block request, and not as a burst request. Then BREQ in GPDMA_CxTR2 must be set for a correct GPDMA execution of the requested peripheral transfer at the hardware level.

1. Depends on which exclusive function is used.

---

### Table 125. Programmed GPDMA1/2 request (continued)

<table>
<thead>
<tr>
<th>GPDMA_CxTR2.REQSEL[7:0]</th>
<th>Selected GPDMA request</th>
</tr>
</thead>
<tbody>
<tr>
<td>109</td>
<td>aes_out_dma</td>
</tr>
<tr>
<td>110</td>
<td>aes_in_dma</td>
</tr>
<tr>
<td>111</td>
<td>hash_in_dma</td>
</tr>
<tr>
<td>112</td>
<td>ucpd1_rx_dma</td>
</tr>
<tr>
<td>113</td>
<td>ucpd1_tx_dma</td>
</tr>
<tr>
<td>114</td>
<td>cordic_read_dma</td>
</tr>
<tr>
<td>115</td>
<td>cordic_write_dma</td>
</tr>
<tr>
<td>116</td>
<td>fmac_read_dma</td>
</tr>
<tr>
<td>117</td>
<td>fmac_write_dma</td>
</tr>
<tr>
<td>118</td>
<td>saes_out_dma</td>
</tr>
<tr>
<td>119</td>
<td>saes_in_dma</td>
</tr>
<tr>
<td>120</td>
<td>i3c1_rx_dma</td>
</tr>
<tr>
<td>121</td>
<td>i3c1_tx_dma</td>
</tr>
<tr>
<td>122</td>
<td>i3c1_tc_dma</td>
</tr>
<tr>
<td>123</td>
<td>i3c1_rs_dma</td>
</tr>
<tr>
<td>124</td>
<td>i2c4_rx_dma</td>
</tr>
<tr>
<td>125</td>
<td>i2c4_tx_dma</td>
</tr>
<tr>
<td>126</td>
<td>Reserved</td>
</tr>
<tr>
<td>127</td>
<td>lptim3_ic1_dma</td>
</tr>
<tr>
<td>128</td>
<td>lptim3_ic2_dma</td>
</tr>
<tr>
<td>129</td>
<td>lptim3_ue_dma</td>
</tr>
<tr>
<td>130</td>
<td>lptim5_ic1_dma</td>
</tr>
<tr>
<td>131</td>
<td>lptim5_ic2_dma</td>
</tr>
<tr>
<td>132</td>
<td>lptim5_ue_dma</td>
</tr>
<tr>
<td>133</td>
<td>lptim6_ic1_dma</td>
</tr>
<tr>
<td>134</td>
<td>lptim6_ic2_dma</td>
</tr>
<tr>
<td>135</td>
<td>lptim6_ue_dma</td>
</tr>
</tbody>
</table>
16.3.6 GPDMA channels with peripheral early termination

A GPDMA channel, if implemented with this feature, can support the early termination of the data transfer from the peripheral which does also support this feature.

Table 127. GPDMA1/2 channel with peripheral early termination

<table>
<thead>
<tr>
<th>GPDMA channel x with peripheral early termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = 0 ) and ( x = 7 )</td>
</tr>
</tbody>
</table>

This GPDMA support is activated when the channel \( x \) is programmed with GPDMA\_CxTR2.PFREQ = 1. Then, the peripheral itself can initiate and request a data transfer completion, before that the GPDMA has transferred the whole block (see Section 16.4.14 for more details).

Table 128. Programmed GPDMA1/2 request with peripheral early termination

<table>
<thead>
<tr>
<th>Programmed GPDMA channel x request with peripheral early termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>i3c1_rx_dma</td>
</tr>
</tbody>
</table>

16.3.7 GPDMA triggers

A GPDMA trigger can be assigned to a GPDMA channel \( x \), via TRIGSEL[5:0] in GPDMA\_CxTR2, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

Table 129. Programmed GPDMA1/2 trigger

<table>
<thead>
<tr>
<th>GPDMA_CxTR2.TRIGSEL[5:0]</th>
<th>Selected GPDMA trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>exti0</td>
</tr>
<tr>
<td>1</td>
<td>exti1</td>
</tr>
<tr>
<td>2</td>
<td>exti2</td>
</tr>
<tr>
<td>3</td>
<td>exti3</td>
</tr>
<tr>
<td>4</td>
<td>exti4</td>
</tr>
<tr>
<td>5</td>
<td>exti5</td>
</tr>
<tr>
<td>6</td>
<td>exti6</td>
</tr>
</tbody>
</table>
Table 129. Programmed GPDMA1/2 trigger (continued)

<table>
<thead>
<tr>
<th>GPDMA_CxTR2.TRIGSEL[5:0]</th>
<th>Selected GPDMA trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>exti7</td>
</tr>
<tr>
<td>8</td>
<td>tamp_trg1</td>
</tr>
<tr>
<td>9</td>
<td>tamp_trg2</td>
</tr>
<tr>
<td>10</td>
<td>tamp_trg3</td>
</tr>
<tr>
<td>11</td>
<td>lptim1_ch1</td>
</tr>
<tr>
<td>12</td>
<td>lptim1_ch2</td>
</tr>
<tr>
<td>13</td>
<td>lptim2_ch1</td>
</tr>
<tr>
<td>14</td>
<td>lptim2_ch2</td>
</tr>
<tr>
<td>15</td>
<td>rtc_alra_trg</td>
</tr>
<tr>
<td>16</td>
<td>rtc_alrb_trg</td>
</tr>
<tr>
<td>17</td>
<td>rtc_wut_trg</td>
</tr>
<tr>
<td>18</td>
<td>gpdma1_ch0_tc</td>
</tr>
<tr>
<td>19</td>
<td>gpdma1_ch1_tc</td>
</tr>
<tr>
<td>20</td>
<td>gpdma1_ch2_tc</td>
</tr>
<tr>
<td>21</td>
<td>gpdma1_ch3_tc</td>
</tr>
<tr>
<td>22</td>
<td>gpdma1_ch4_tc</td>
</tr>
<tr>
<td>23</td>
<td>gpdma1_ch5_tc</td>
</tr>
<tr>
<td>24</td>
<td>gpdma1_ch6_tc</td>
</tr>
<tr>
<td>25</td>
<td>gpdma1_ch7_tc</td>
</tr>
<tr>
<td>26</td>
<td>gpdma2_ch0_tc</td>
</tr>
<tr>
<td>27</td>
<td>gpdma2_ch1_tc</td>
</tr>
<tr>
<td>28</td>
<td>gpdma2_ch2_tc</td>
</tr>
<tr>
<td>29</td>
<td>gpdma2_ch3_tc</td>
</tr>
<tr>
<td>30</td>
<td>gpdma2_ch4_tc</td>
</tr>
<tr>
<td>31</td>
<td>gpdma2_ch5_tc</td>
</tr>
<tr>
<td>32</td>
<td>gpdma2_ch6_tc</td>
</tr>
<tr>
<td>33</td>
<td>gpdma2_ch7_tc</td>
</tr>
<tr>
<td>34</td>
<td>tim2_trgo</td>
</tr>
<tr>
<td>35</td>
<td>tim15_trgo</td>
</tr>
<tr>
<td>36</td>
<td>tim12_trgo</td>
</tr>
<tr>
<td>37</td>
<td>lptim3_ch1</td>
</tr>
<tr>
<td>38</td>
<td>lptim3_ch2</td>
</tr>
<tr>
<td>39</td>
<td>lptim4_ait</td>
</tr>
<tr>
<td>40</td>
<td>lptim5_ch1</td>
</tr>
<tr>
<td>41</td>
<td>lptim5_ch2</td>
</tr>
</tbody>
</table>
16.4 GPDMA functional description

16.4.1 GPDMA block diagram

Figure 66. GPDMA block diagram

16.4.2 GPDMA channel state and direct programming without any linked-list

After a GPDMA reset, a GPDMA channel x is in idle state. When the software writes 1 in GPDMA_CxCR.EN, the channel takes into account the value of the different channel configuration registers (GPDMA_CxXXX), switches to the active/non-idle state and starts to execute the corresponding requested data transfers.
After enabling/starting a GPDMA channel transfer by writing 1 in GPDMA_CxCR.EN, a GPDMA channel interrupt on a complete transfer notifies the software that the GPDMA channel is back in idle state (EN is then deasserted by hardware) and that the channel is ready to be reconfigured then enabled again.

The figure below illustrates this GPDMA direct programming without any linked-list (GPDMA_CxLLR = 0).

**Figure 67. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0)**

16.4.3 **GPDMA channel suspend and resume**

The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master ports. Then the software can observe, in a steady state, any read register or register field that is hardware modifiable.

Note that an ongoing GPDMA transfer can be a data transfer (a source/destination burst transfer) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to GPDMA_CxCR.SUSP.

**Figure 68. GPDMA channel suspend and resume sequence**

![Diagram of GPDMA channel suspend and resume sequence]

*Note: A suspend and resume sequence does not impact the GPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.*

### 16.4.4 GPDMA channel abort and restart

Alternatively, like for aborting a continuous GPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master port.
3. The software resets the channel by writing 1 to GPDMA_CxCR.RESET. This causes
the reset of the FIFO, the reset of the channel internal state, the reset of the
GPDMA_CxCR.EN bit, and the reset of the GPDMA_CxCR.SUSP bit.

4. The software safely reconfigures the channel. The software must reprogram the
hardware-modified GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR
registers.

5. In order to restart the aborted then reprogrammed channel, the software enables it
again by writing 1 to the GPDMA_CxCR.EN bit.

**Figure 69. GPDMA channel abort and restart sequence**

16.4.5 **GPDMA linked-list data structure**

Alternatively to the direct programming mode, a channel can be programmed by a list of
transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.
The base address in memory of the data structure of a next LLIn+1 of a channel x is the sum of the following:

- the link base address of the channel x (in GPDMA_CxLBAR)
- the link address offset (LA[15:2] field in GPDMA_CxLLR). The linked-list register GPDMA_CxLLR is the updated result from the data structure of the previous LLIn of the channel x.

The data structure for each LLI may be specific.

A linked-list data structure is addressed following the value of the UT1, UT2, UB1, USA, UDA and ULL bits, plus UB2 and UT3, in GPDMA_CxLLR.

In linked-list mode, each GPDMA linked-list register (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3 or GPDMA_CxBR2) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of the GPDMA_CxLLR register that was conditionally updated from the linked-list data structure of the previous LLI.

**Static linked-list data structure**

For example, when the update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3) in GPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- channel x (x = 0 to 5) contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR (see Figure 70) and including the first linked-list register file (LLI0) and the next LLIs (such as LLI1, LLI2) in the memory
- channel x(x = 6 to 7), contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2 (see Figure 71), and including the first linked-list register file (LLI0) and the next LLIs (such as LLI1, LLI2) in the memory
Figure 70. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x
Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of the register GPDMA_CxLLR is pointing to the updated value of the GPDMA_CxTR2 instead of the GPDMA_CxTR1 which is not to be modified (see Figure 72).

Example: if UT1 = UB1 = USA = 0 and if UT3 = UB2 = 0, when channel x is with 2D addressing, and if UT2 = UDA = ULL = 1, the next LLI does not contain an (updated) value for GPDMA_CxTR1, nor GPDMA_CxBR1, nor GPDMA_CxSAR, nor GPDMA_CxDar, nor GPDMA_CxTR3, nor GPDMA_CxBR2 when channel x is with 2D addressing. The next LLI contains an updated value for GPDMA_CxTR2, GPDMA_CxDAR, and GPDMA_CxLLR, as shown in Figure 73.
The user must program GPDMA_CxLLR for each LLI_n to be 32-bit aligned and not to exceed the 64-Kbyte addressable space pointed by GPDMA_CxLBAR.

### 16.4.6 Linked-list item transfer execution

A LLI_n transfer is the sequence of:

1. a data transfer: GPDMA executes the data transfer as described by the GPDMA internal register file (this data transfer can be void/null for LLI_0)
2. a conditional link transfer: GPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI_{n+1}, as defined by the GPDMA_CxLLR value of the LLI_n.

**Note:** The initial data transfer as defined by the internal register file (LLI_0) can be null (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxTR2.PFREQ = 0) provided that the conditional update bit UB1 in GPDMA_CxLLR is set (meaning there is a non-null data transfer described by the next LLI in the memory to be executed).

Depending on the intended GPDMA usage, a GPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, GPDMA_CxCR.LSM = 0) or a GPDMA channel x can be programmed for a single execution of a LLI (link step mode, GPDMA_CxCR.LSM = 1), as described in the next sections.
16.4.7 GPDMA channel state and linked-list programming in run-to-completion mode

When GPDMA_CxCR.LSM = 0 (in full list execution mode, execution of the full sequence of LLIs, named run-to-completion mode), a GPDMA channel x is initially programmed, started by writing 1 to GPDMA_CxCR.EN, and after completed at channel level. The channel transfer is:

- configured with at least the following:
  - the first LLI₀, internal linked-list register file: GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2
  - the last LLIₙ₋₁, described by the linked-list data structure in memory, as defined by the GPDMA_CxLLR reflecting the before last LLIₙ₋₂

- completed when GPDMA_CxLLR[31:0] = 0, GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLIₙ₋₁ transfer

GPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion and means the following:

- The 16 low significant bits GPDMA_CxLLR.LA[15:0] of the next link address are null.
- All the update bits GPDMA_CxLLR.Uxx are null (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3).

The channel may never be completed when GPDMA_CxLLR.LSM = 0:

- If the last LLIₙ₋₁ is recursive, pointing to itself as a next LLI:
  - either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated by the same value
  - or GPDMA_CxLLR.ULL = 0
- If LLIₙ₋₁ is pointing to a previous LLI

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see Section 16.4.14 for more details).

In the typical run-to-completion mode, the allocation of a GPDMA channel, including its fine programming, is done once during the GPDMA initialization. In order to have a reserved data communication link and GPDMA service during run-time, for continuously repeated transfers (from/to a peripheral respectively to/from memory or for memory-to-memory transfers). This reserved data communication link can consist of a channel, or the channel can be shared and a repeated transfer consists of a sequence of LLIs.

Figure 74 depicts the GPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 74 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

In Figure 74, BNDT ≠ 0 is the typical condition for starting the first data transfer in this figure. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 16.3.6).
Figure 74. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0)

Channel state = Idle

- Initialize DMA channel
- Enable DMA channel
- Reconfigure DMA channel

Channel state = Active

- Valid user setting?
  - Y: Setting USEF = 1, Disabling DMA channel
  - N: BNDT # 0?
    - Y: Setting USEF = 1, Disabling DMA channel
    - N: Executing once the data transfer from the register file
      - No transfer error?
        - Y: Setting USEF = 1, Disabling DMA channel
        - N: LLR # 0?
          - Y: Loading next LLI into the register file
          - N: Setting USEF = 1, Disabling DMA channel
          - N: Setting TCF = 1, Disabling DMA channel
          - Y: Setting USEF = 1, Disabling DMA channel
          - N: Setting ULEF = 1, Disabling DMA channel

- Setting DTEF = 1, Disabling DMA channel

End
Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following fields of the GPDMA_CxTR2 in the data structure of the LLI_{n-1}:

- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[5:0] (see Section 16.3.7 for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory and to generate a transfer complete event gpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

The figure below depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

**Caution:** This use case is restricted to an application with a LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.
16.4.8 GPDMA channel state and linked-list programming in link step mode

When GPDMA_CxCR.LSM = 1 (in link step execution mode, single execution of one LLI), a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

A GPDMA channel transfer can be programmed at LLI level, started by writing 1 into GPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI\(_n\) transfer is described with:
  - GPDMA_CxTR1 defines the source/destination elementary single/burst transfers.
  - GPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]) and, for channel x (x = 6 to 7), the number of blocks at a 2D/repeated block level (BRC[10:0]+1) and the incrementing/decrementing mode for address offsets.
- GPDMA_CxTR2 defines the input control (request, trigger) and the output control (transfer complete event) of the transfer.
- GPDMA_CxSAR/GPDMA_CxDAR define the source/destination transfer start address.
- GPDMA_CxTR3 for channel x (x = 6 to 7) defines the source/destination additional address offset between burst transfers.
- GPDMA_CxBR2 for channel x (x = 6 to 7) defines the source/destination additional address offset between blocks at a 2D/repeated block level.
- GPDMA_CxLLR defines the data structure and the address offset of the next LLI_{n+1} in the memory.

- The current LLI_{n} transfer is completed after the single execution of the current LLI_{n}:
  - after the (conditional) data transfer completion (when GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0
  - after the (conditional) update of the GPDMA link register file from the data structure of the next LLI_{n+1} in memory

**Note:** If a LLI is recursive (pointing to itself as a next LLI, either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated by the same value, or GPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see Section 16.4.14 for more details).

The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_{n} (all update bits of GPDMA_CxLLR have a static value, LLI_{n}.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Figure 76 depicts the GPDMA channel execution mode, and its programming in link step mode.

**Note:** Figure 76 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 76, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 16.3.6).
Figure 76. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1)

Channel state = Idle
- Initialize DMA channel
- Enable DMA channel
- Reconfigure DMA channel

Channel state = Active
- Valid user setting?
  - Y: Setting TCF = 1, Disabling DMA channel
  - N: No transfer error?
    - Y: Setting USEF = 1, Disabling DMA channel
    - N: Setting USEF = 1, Disabling DMA channel
- BNDT ≠ 0?
  - Y: Setting USEF = 1, Disabling DMA channel
  - N: Executing once the data transfer from the register file
- No transfer error?
  - Y: Setting USEF = 1, Disabling DMA channel
  - N: LLR ≠ 0?
    - Y: Setting USEF = 1, Disabling DMA channel
    - N: Loading next LLI into the register file
- No transfer error?
  - Y: Setting USEF = 1, Disabling DMA channel
  - N: Valid user setting?
    - Y: Setting USEF = 1, Disabling DMA channel
    - N: Setting USEF = 1, Disabling DMA channel
- End
Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after GPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in the figure below.

**Figure 77. Building LLI_{n+1}: GPDMA dynamic linked-lists in link step mode**

Run-time replacing a LLI_n with a new LLI'_n in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new LLI'_n, after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory by overwriting directly the linked-list register file with the new LLI'_n, as shown in the figure below.
Run-time replacing a LLI_n with a new LLI_n' in link step mode (in the memory)

The software can build and insert a new LLI_n' and LLI_{n+1}' in the memory, after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new LLI_{n'}) as shown in the figure below.
Figure 79. Replace with a new LLI\textsubscript{n} and LLI\textsubscript{n+1} in memory in link step mode (option 1)

LSM = 1 with 1-stage linked-list programming:
Overwriting the (pre)loaded LLI\textsubscript{n} linked-list register file with a new LLI\textsubscript{n} and LLI\textsubscript{n+1} in memory and overwrite partly linked-list register file
(DMA\_CxBR1.BNDT = 0 and DMA\_CxLLR to point to new LLI\textsubscript{n})
DMA executes LLI\textsubscript{n-1} and load LLI\textsubscript{n} then CPU builds (LLI\textsubscript{n} and LLI\textsubscript{n+1}) and overwrite (BR1 and LLR)

DMA Channel

CPU

Execution LLI\textsubscript{n-1} data transfer

Loading LLI\textsubscript{n}

Transfer complete interrupt

Build LLI\textsubscript{n} and LLI\textsubscript{n+1} in memory

Write DMA\_CxBR1.BNDT = 0
Write DMA\_CxLLR to point to new LLI\textsubscript{n}

Enable DMA channel

Loading LLI\textsubscript{n-1}

Transfer complete interrupt

Enable DMA channel

Execution LLI\textsubscript{n+1} data transfer

Loading LLI\textsubscript{n+1}

Transfer complete interrupt
**Run-time replacing a LLI_{n} with a new LLI'_{n} in link step mode**

Other software implementations exist. Meanwhile GPDMA executes the transfer from the LLI_{n-1} and loads a formerly elaborated LLI_{n} from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new LLI'_{n} and a new LLI'_{n+1}.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for LLI_{n-1} is then restricted to overwrite GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new LLI'_{n}, as shown in the figure below.

**Figure 80. Replace with a new LLI'_{n} and LLI'_{n+1} in memory in link step mode (option 2)**

LSM = 1 with 1-stage linked-list programming:

Overwriting the (pre)loaded LLI_{n} linked-list register file by building new LLI'_{n} and LLI'_{n+1} in memory while disabling (temporary) channel interrupt at NVIC level, and overwriting DMA_CxBR1.BNDT = 0 and DMA_CxLLR to point to new LLI'_{n}.

DMA executes LLI_{n-1} and loading LLI_{n} while CPU builds (LLI'_{n} and LLI'_{n+1}), then CPU overwrites (BR1 and LLR).
16.4.9 GPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (GPDMA_CxCR.EN = 0) and update the execution mode (GPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

In any execution mode, the software can:

- reprogram LLI\_n+1 in the memory to finally complete the channel by this LLI\_n+1 (clear the GPDMA_CxLLR of this LLI\_n+1), before that this LLI\_n+1 is loaded/used by the GPDMA channel
- abort and reconfigure the channel with a LSM update (see Section 16.4.4.)

In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI\_n-1.

*Figure 81* shows the overall and unified GPDMA linked-list programming, whatever is the execution mode.

---

**Note:** *Figure 81* is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

*In Figure 81, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 16.3.6).*
Figure 81. GPDMA channel execution and linked-list programming

Channel state = Idle

- Initialize DMA channel
- Enable DMA channel
- Reconfigure DMA channel

Channel state = Active

- Valid user setting?
  - Yes
    - Setting USEF = 1
    - Disabling DMA channel
  - No
    - BNDT ≠ 0?
      - Yes
        - Executing once the data transfer from the register file
      - No
        - No transfer error?
          - Yes
            - Setting DTEF = 1
            - Disabling DMA channel
          - No
            - LLR ≠ 0?
              - Yes
                - Loading next LLI into the register file
              - No
                - No transfer error?
                  - Yes
                    - Setting ULEF = 1
                    - Disabling DMA channel
                  - No
                    - Valid user setting?
                      - Yes
                        - Setting USEF = 1
                        - Disabling DMA channel
                      - No
                        - LSM = 1?
                          - Yes
                            - Setting TCF = 1
                            - Disabling DMA channel
                          - No
                            - End

- Setting TCF = 1
  - Disabling DMA channel
16.4.10 GPDMA FIFO-based transfers

There is a single transfer operation mode: the FIFO mode. There are FIFO-based transfers. Any channel \( x \) is implemented with a dedicated FIFO whose size is defined by \( \text{dma_fifo_size}[x] \) (see Section 16.3.2 for more details).

GPDMA burst

A programmed transfer at the lowest level is a GPDMA burst.

A GPDMA burst is a burst of data received from the source, or a burst of data sent to the destination. A source (and destination) burst is programmed with a burst length by the field \( \text{SBL}_1[5:0] \) (respectively \( \text{DBL}_1[5:0] \)), and with a data width defined by the field \( \text{SDW_LOG2}[1:0] \) (respectively \( \text{DDW_LOG2}[1:0] \)) in the \( \text{GPDMA_CxTR1} \) register.

The addressing mode after each data (named beat) of a GPDMA burst is defined by \( \text{SINC} \) and \( \text{DINC} \) in \( \text{GPDMA_CxTR1} \), for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of a GPDMA source/destination burst (defined by \( \text{GPDMA_CxSAR} \) and \( \text{GPDMA_CxDAR} \)) must be aligned with the respective data width.

The table below lists the main characteristics of a GPDMA burst.

<table>
<thead>
<tr>
<th>SDW_LOG2[1:0]</th>
<th>Data width (bytes)</th>
<th>SINC/DINC</th>
<th>SBL_1[5:0]</th>
<th>Burst length (data/beats)</th>
<th>Next data/beat address</th>
<th>Next burst address</th>
<th>Burst address alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>0 (fixed)</td>
<td></td>
<td></td>
<td>+ 0</td>
<td>+ 0</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>+ 1</td>
<td>+ (n + 1)</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>+ 2</td>
<td>+ 2 * (n + 1)</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>+ 4</td>
<td>+ 4 * (n + 1)</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>forbidden user setting, causing USEF generation and none burst to be issued.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. When \( \text{S/DBL}_1[5:0] = 0 \), burst is of length 1. Then burst can be also named as single.

The next burst address in the above table is the next source/destination default address pointed by \( \text{GPDMA_CxSAR} \) or \( \text{GPDMA_CxDAR} \), once the programmed source/destination burst is completed. This default value refers to the fixed/contiguously incremented address.
GPDMA burst with 2D addressing (channel x = 6, 7)

When the channel has additional 2D addressing feature, this default value refers to the value without taking into account the two programmed incremented or decremented offsets. These two additional offsets (with a null default value) are applied:

- after each completed source/destination burst, as defined respectively by GPDMA_CxTR2.SAO[12:0]/DAO[12:0] and GPDMA_CxBR1.SDEC/DDEC
- after each completed block, as defined respectively by GPDMA_CxBR2.BRSAO[15:0]/BRDAO[15:0] and GPDMA_CxBR1.BRSDEC/BRDDEC

Then, a 2D/repeated block can be addressed with a first programmed address jump after each completed burst, and with a second programmed address jump after each block, as depicted by the figure below with a 2D destination buffer.
Figure 82. Programmed 2D addressing

Programmable address jumps 1) after burst and 2) after block.
Example:
burst: \( l \times \) words (DBL \(_1\)=I-1; DDW LOG2='b10)
block: \( J \times \) bursts (BNDT=J**4)
LLI: \( K \times \) blocks (BRC=K-1)
GPDMA FIFO-based burst

In FIFO-mode, a transfer generally consists of two pipelined and separated burst transfers:

- one burst from the source to the FIFO over the allocated source master port, as defined by GPDMA_CxTR1.SAP
- one burst from the FIFO to the destination over the allocated destination master port, as defined by GPDMA_CxTR1.DAP

GPDMA source burst

The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to get one new burst from the source):

when FIFO level ≤ 2^{dma_fifo_size[x]} - (SBL_1[5:0]+1) * 2^{SDW_LOG2[1:0]}

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- 2^{dma_fifo_size[x]} is the half of the FIFO size of the channel x, in bytes (see Section 16.3.2 for the implementation details and dma_fifo_size[x] value).
- (SBL_1[5:0]+1) * 2^{SDW_LOG2[1:0]} is the size of the programmed source burst transfer, in bytes.

Based on the channel priority (GPDMA_CxCR.PRIOR[1:0]), this ready FIFO-based source transfer is internally arbitrated versus the other requested and active channels.

GPDMA destination burst

The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to push one new burst to the destination):

when FIFO level ≥ (DBL_1[5:0]+1) * 2^{DDW_LOG2[1:0]}

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- (DBL_1[5:0]+1) * 2^{DDW_LOG2[1:0]} is the size of the programmed destination burst transfer, in bytes.

Based on the channel priority, this ready FIFO-based destination transfer is internally arbitrated versus the other requested and active channels.

GPDMA burst vs source block size, 1-Kbyte address boundary and FIFO size

The programmed source/destination GPDMA burst is implemented with an AHB burst as is, unless one of the following conditions is met:

- When half of the FIFO size of the channel x is lower than the programmed source/destination burst size, the programmed source/destination GPDMA burst is implemented with a series of singles or bursts of a lower size, each transfer being of a size that is lower or equal than half of the FIFO size, without any user constraint.
- if the source block size (GPDMA_CxBR1.BNDT[15:0]) is not a multiple of the source burst size but is a multiple of the data width of the source burst (GPDMA_CxTR1.SDW_LOG2[1:0]), the GPDMA modifies and shortens bursts into singles or bursts of lower length, in order to transfer exactly the source block size, without any user constraint.
• if the source/destination burst transfer have crossed the 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.

• If the source/destination burst length exceeds 16 on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.

In any case, the GPDMA keeps ensuring source/destination data (and address) integrity without any user constraint. The current FIFO level (software readable in GPDMA_CxSR) is compared to and updated with the effective transfer size, and the GPDMA re-arbitrates between each AHB single or burst transfer, possibly modified.

Based on the channel priority, each single or burst of a lower burst size versus the programmed burst, is internally arbitrated versus the other requested and active channels. 

**Note:** In linked-list mode, the GPDMA read transfers related to the update of the linked-list parameters from the memory to the internal GPDMA registers, are scheduled over the link allocated port, as programmed by GPDMA_CxCR.LAP.

**GPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension and left/right alignment**

The data handling is controlled by GPDMA_CxTR1. The source/destination data width of the programmed burst is byte, half-word or word, as per the SDW_LOG2[21:0] and DDW_LOG2[1:0] fields (see Table 131).

The user can configure the data handling between transferred data from the source and transfer to the destination. More specifically, programmed data handling is orderly performed with:

1. **Byte-based source reordering**
   - If SBX = 1 and if source data width is a word, the two bytes of the unaligned half-word at the middle of each source data word are exchanged.

2. Data width conversion by packing, unpacking, padding or truncation, if destination data width is different than the source data width, depending on PAM[1:0]:
   - If destination data width > source data width, the post SBX source data is either right-aligned and padded with 0 s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
   - If destination data width < source data width, the post SBX data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.

3. **Byte-based destination re-ordering:**
   - If DBX = 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM[1:0] half-words.
   - If DHX = 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM[1:0] words. 

**Note:** Left-alignment with 0s-padding can be achieved by programming both a right-alignment with a 0s-padding and a destination byte-based re-ordering.
The table below lists the possible data handling from the source to the destination.

**Table 131. Programmed data handling**

<table>
<thead>
<tr>
<th>SDW_LOG2 [1:0]</th>
<th>Source data</th>
<th>Source data stream(1)</th>
<th>SB X</th>
<th>DDW_LOG2 [1:0]</th>
<th>Destination data</th>
<th>PAM<a href="2">1:0</a></th>
<th>DB X</th>
<th>DX</th>
<th>Destination data stream(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>00</td>
<td>00 (RA, 0P)</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>B7,B6,B5,B4,B3,B2,B1,B0</td>
</tr>
<tr>
<td>00</td>
<td>Byte</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0B3,0B2,0B1,0B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B3,B2,B1,B0,0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>01 (RA, SE)</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>SB3,SB2,SB1,SB0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B3,B2,B1,B0,S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>B7,B6,B4,B3,B2,B1,B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>B7,B6,B5,B4,B3,B2,B1,B0</td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td></td>
<td>00</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>000B1,000B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00B1,000B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>0B1,000B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>B1,000B0</td>
</tr>
<tr>
<td>10</td>
<td>Word</td>
<td></td>
<td>00</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>SSSB1,SSSB0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SSB1,SSS&lt;sub&gt;B0&lt;/sub&gt;S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>SB1,SSSB&lt;sub&gt;B0&lt;/sub&gt;SS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>B1,SSS&lt;sub&gt;B0&lt;/sub&gt;SS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>B7,B6,B5,B4,B3,B2,B1,B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>B7,B6,B5,B4,B3,B2,B1,B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>B5,B4,B7,B6,B3,B2,B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>B5,B4,B7,B6,B3,B2,B1</td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td></td>
<td>00</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>B6,B5,B4,B2,B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B6,B5,B4,B2,B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>B7,B6,B5,B4,B3,B2,B1,B0</td>
</tr>
</tbody>
</table>
### Table 131. Programmed data handling (continued)

<table>
<thead>
<tr>
<th>SDW_LOG2 [1:0]</th>
<th>Source data</th>
<th>Source data stream(^{(1)})</th>
<th>SB X</th>
<th>DDW_LOG2 [1:0]</th>
<th>Destination data</th>
<th>PAM[1:0](^{(2)})</th>
<th>DB X</th>
<th>DH X</th>
<th>Destination data stream(^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Half-word</td>
<td>B(_7)B(_6)B(_5)B(_4), B(_3)B(_2), B(_1)B(_0)</td>
<td>x</td>
<td>10</td>
<td>Word</td>
<td>00 (RA, 0P)</td>
<td>0</td>
<td>0</td>
<td>B(_7)B(_6)B(_5)B(_4)B(_3)B(_2)B(_1)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>B(_8)B(_7)B(_5)B(_4)B(_2)B(_0)B(_1)B(_0)</td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td>B(_7)B(_6)B(_5)B(_4), B(_3)B(_2), B(_1)B(_0)</td>
<td>x</td>
<td>00</td>
<td>Byte</td>
<td>00 (RA, LT)</td>
<td>x</td>
<td></td>
<td>B(_1)B(_2)B(_3)B(_4)B(_0)B(_1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 (LA, RT)</td>
<td></td>
<td></td>
<td>B(_1)B(_2)B(_3)B(_4)B(_0)B(_1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10 (UNPACK)</td>
<td></td>
<td></td>
<td>B(_1)B(_2)B(_3)B(_4)B(_0)B(_1)</td>
</tr>
<tr>
<td>10</td>
<td>Word</td>
<td>B(_7)B(_6)B(_5)B(_4), B(_3)B(_2), B(_1)B(_0)</td>
<td>0</td>
<td>00</td>
<td>Byte</td>
<td>00 (RA, LT)</td>
<td>0</td>
<td>1</td>
<td>B(_7)B(_6)B(_5)B(_4)B(_3)B(_2)B(_1)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 (LA, RT)</td>
<td></td>
<td></td>
<td>B(_7)B(_6)B(_5)B(_4)B(_3)B(_2)B(_1)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10 (UNPACK)</td>
<td></td>
<td></td>
<td>B(_7)B(_6)B(_5)B(_4)B(_3)B(_2)B(_1)B(_0)</td>
</tr>
</tbody>
</table>

\(^{(1)}\) Data stream:\n- SDW\_LOG2 [1:0] represents the source data width.
- DDW\_LOG2 [1:0] represents the destination data width.

\(^{(2)}\) PAM[1:0] controls the memory access pattern.
- DB X: Data byte selection
- DH X: Data half-word selection
### 16.4.11 GPDMA transfer request and arbitration

#### GPDMA transfer request

As defined by GPDMA_CxTR2, a programmed GPDMA data transfer is requested with one of the following:

- **a software request if the control bit SWREQ = 1**: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- **an input hardware request coming from a peripheral if SWREQ = 0**: The selection of the GPDMA hardware peripheral request is driven by the REQSEL[7:0] field (see Section 16.3.4). The selected hardware request can be one of the following:
  - an hardware request from a peripheral configured in GPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - an hardware request from a peripheral for its control registers update from the memory
  - an hardware request from a peripheral for a read of its status registers transferred to the memory

---

### Table 131. Programmed data handling (continued)

<table>
<thead>
<tr>
<th>SDW_LOG2 [1:0]</th>
<th>Source data</th>
<th>Source data stream(^{(1)})</th>
<th>SB X</th>
<th>DDW_LOG2 [1:0]</th>
<th>Destination data</th>
<th>PAM[1:0](^{(2)})</th>
<th>DB X</th>
<th>DH X</th>
<th>Destination data stream(^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Word</td>
<td>B7B6B5B4, B3B2B1B0</td>
<td>0</td>
<td>10</td>
<td>Word</td>
<td>xx</td>
<td>0</td>
<td>0</td>
<td>B7B6B5B4, B3B2B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>00</td>
<td>Byte</td>
<td>00 (RA, LT)</td>
<td>1</td>
<td>B6B5B4, B3B2B1B0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>00</td>
<td>Byte</td>
<td>01 (LA, RT)</td>
<td>1</td>
<td>B15B14B13B12B11B10B9B8B7B6B5B4B3B2B1B0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>01</td>
<td>Half-word</td>
<td>00 (RA, LT)</td>
<td>0</td>
<td>B7B6B5B4, B3B2B1B0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>01</td>
<td>Half-word</td>
<td>01 (LA, RT)</td>
<td>1</td>
<td>B7B6B5B4, B3B2B1B0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>01</td>
<td>Half-word</td>
<td>1x (UNPACK)</td>
<td>0</td>
<td>B7B6B5B4, B3B2B1B0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>01</td>
<td>Half-word</td>
<td>1x (UNPACK)</td>
<td>1</td>
<td>B7B6B5B4, B3B2B1B0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>10</td>
<td>Word</td>
<td>xx</td>
<td>1</td>
<td>B6B5B4, B3B2B1B0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>10</td>
<td>Word</td>
<td>xx</td>
<td>1</td>
<td>B6B5B4, B3B2B1B0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>10</td>
<td>Word</td>
<td>xx</td>
<td>0</td>
<td>B6B5B4, B3B2B1B0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>10</td>
<td>Word</td>
<td>xx</td>
<td>1</td>
<td>B6B5B4, B3B2B1B0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>10</td>
<td>Word</td>
<td>xx</td>
<td>0</td>
<td>B6B5B4, B3B2B1B0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>10</td>
<td>Word</td>
<td>xx</td>
<td>1</td>
<td>B6B5B4, B3B2B1B0</td>
<td></td>
</tr>
</tbody>
</table>

1. Data stream is timely ordered starting from the byte with the lowest index (B0).
2. RA = right aligned, LA = left aligned, RT = right truncated, LT = left truncated, 0P = zero bit padding up to the destination data width, SE = sign bit extended up to the destination data width.

---
Caution: The user must not assign a same input hardware peripheral GPDMA request via GPDMA_CxTR.REQSEL[7:0] to two different channels, if at a given time this request is asserted by the peripheral and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

GPDMA transfer request for arbitration

A ready FIFO-based GPDMA source single/burst transfer (from the source address to the FIFO) to be scheduled over the allocated master port (GPDMA_CxTR1.SAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

A ready FIFO-based GPDMA destination single/burst transfer (from the FIFO to the destination address) to be scheduled over the allocated master port (GPDMA_CxTR1.DAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

An arbitrated GPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in memory to one of the linked-list registers (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3, GPDMA_CxBR2). Each 32-bit read from memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the allocated master port (GPDMA_CxCR.LAP).

Whatever the requested data transfer is programmed with a software request for a memory-to-memory transfer (GPDMA_CxTR2.SWREQ = 1), or with a hardware request (GPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer and whatever is the hardware request type, re-arbitration occurs after each granted single/burst transfer.

When an hardware request is programmed from a destination peripheral (GPDMA_CxTR2.SWREQ = 0 and GPDMA_CxTR2.DREQ = 1), the first memory read of a (possibly 2D/repeated) block (the first ready FIFO-based source burst request), is gated by the occurrence of the corresponding and selected hardware request. This first read request to memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

GPDMA arbitration

The GPDMA arbitration is directed from the 4-grade assigned channel priority (GPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in Figure 83, is defined by:

- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels
  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single/burst transfers are round-robin arbitrated.

- three low-priority traffic classes (queues 0, 1 or 2) for non time-sensitive channels with priority 0, 1 or 2
  Each requested single/burst transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.
Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.

Any queue 0, 1 or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.

Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1 and queue 2.

Figure 83. GPDMA arbitration policy

GPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- Equal maximum bandwidth between requests with same priority
- Reserved bandwidth (noted as \(B_{Q3}\)) to the time-sensitive requests (with priority 3)
- Residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive GPDMA requests: req0 and req1. There are the following programming possibilities:
  - If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (\(B_{req0}\)) is equal to the allocated bandwidth to req1 (\(B_{req1}\)).
    \[
    B_{req0} = B_{req1} = \frac{1}{2} * (1 - B_{Q3})
    \]
  - If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (\(B_{P0}\)) is 3 times less than the allocated bandwidth to req1 (\(B_{P1}\)).
    \[
    B_{req0} = B_{P0} = \frac{1}{2} * \frac{1}{2} * (1 - B_{Q3}) = \frac{1}{4} * (1 - B_{Q3})
    \]
    \[
    B_{req1} = B_{P1} = \frac{(1/2 + 1)}{1/3} * (1 - B_{Q3}) = \frac{3}{4} * (1 - B_{Q3})
    \]
  - If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (\(B_{P0}\)) is 5 times less than the allocated bandwidth to req1 (\(B_{P2}\)).
    \[
    B_{req0} = B_{P0} = \frac{1}{2} * 1/3 * (1 - B_{Q3}) = \frac{1}{6} * (1 - B_{Q3})
    \]
    \[
    B_{req1} = B_{P2} = \frac{(1/2 + 1 + 1)}{1/3} * (1 - B_{Q3}) = \frac{5}{6} * (1 - B_{Q3})
    \]

The above computed bandwidth calculation is based on a theoretical input request, always active for any GPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one GPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.
• **Example 2**: application where the user distributes a same non-null \(N\) number of GPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0 \((B_{P0})\) is
    \[
    B_{P0} = \frac{1}{3} \times \frac{1}{3} \times (1 - B_{Q3}) = \frac{1}{9} \times (1 - B_{Q3})
    \]
  - The allocated bandwidth to the set of requests of priority 1 \((B_{P1})\) is
    \[
    B_{P1} = (\frac{1}{3} + \frac{1}{2}) \times \frac{1}{3} \times (1 - B_{Q3}) = \frac{5}{18} \times (1 - B_{Q3})
    \]
  - The allocated bandwidth to the set of requests of priority 2 \((B_{P2})\) is
    \[
    B_{P2} = (\frac{1}{3} + \frac{1}{2} + 1) \times \frac{1}{3} \times (1 - B_{Q3}) = \frac{11}{18} \times (1 - B_{Q3})
    \]
  - The allocated bandwidth to any request \(n\) \((B_{n})\) among the \(N\) requests of that priority \(Pi\) \((i = 0 \text{ to } 2)\) is
    \[
    B_n = \frac{1}{N} \times B_{Pi}
    \]
  - The allocated bandwidth to any request \(n\) of priority 0 \((B_{n, 0})\) is
    \[
    B_{n, 0} = \frac{1}{N} \times \frac{1}{9} \times (1 - B_{Q3})
    \]
    \[
    B_{n, 1} = \frac{1}{N} \times \frac{5}{18} \times (1 - B_{Q3})
    \]
    \[
    B_{n, 2} = \frac{1}{N} \times \frac{11}{18} \times (1 - B_{Q3})
    \]
  
  In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:
• \(I\) requests \((I \geq 0)\) assigned to priority 0
  - If \(I > 0\), these requests are noted from \(i = 0\) to \(I-1\).
• \(J\) requests \((J \geq 0)\) assigned to priority 1
  - If \(J > 0\), these requests are noted from \(j = 0\) to \(J-1\).
• \(K\) requests \((K > 0)\) assigned to priority 2
  - These requests are noted from \(k = 0\) to \(K-1\)
• \(L\) requests \((L \geq 0)\) assigned to priority 3
  - If \(L > 0\), these requests are noted from \(l = 0\) to \(L-1\).

As \(B_{Q3}\) is the reserved bandwidth to time-sensitive requests, the bandwidth for each request \(L\) with priority 3 is:
• \(B_l = B_{Q3} / L\) for \(L > 0\) (else: \(B_l = 0\))

The bandwidth for each non-time sensitive queue is:
• \(B_{Q0} = \frac{1}{3} \times (1 - B_{Q3})\)
• \(B_{Q1} = \frac{1}{3} \times (1 - B_{Q3})\)
• \(B_{Q2} = \frac{1}{3} \times (1 - B_{Q3})\)

The bandwidth for the set of requests with priority 0 is:
• \(B_{P0} = I / (I + J + K) \times B_{Q0}\)

The bandwidth for each request \(i\) with priority 0 is:
• \(B_i = B_{P0} / I\) for \(L > 0\) (else \(B_{P0} = 0\))

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:
• \(B_{P1,Q0} = J / (I + J + K) \times B_{Q0}\)
The bandwidth for the set of requests with priority 1 and routed to queue 1 is:
• \( B_{P1,Q1} = \frac{J}{J + K} \cdot B_{Q1} \)

The total bandwidth for the set of requests with priority 1 is:
• \( B_{P1} = B_{P1,Q0} + B_{P1,Q1} \)

The bandwidth for each request \( j \) with priority 1 is:
• \( B_j = \frac{B_{P1}}{J} \) for \( J > 0 \) (else \( B_j = 0 \))

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:
• \( B_{P2,Q0} = \frac{K}{I + J + K} \cdot B_{Q0} \)

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:
• \( B_{P2,Q1} = \frac{K}{J + K} \cdot B_{Q1} \)

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:
• \( B_{P2,Q2} = B_{Q2} \)

The total bandwidth for the set of requests with priority 2 is:
• \( B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2} \)

The bandwidth for each request \( k \) with priority 2 is:
• \( B_k = \frac{B_{P2}}{K} \) (\( K > 0 \) in the general case)

Thus finally the maximum allocated residual bandwidths for any \( i, j, k \) non-time sensitive request are:
• in the general case (when there is at least one request \( k \) with a priority 2 (\( K > 0 \))):
  – \( B_i = \frac{1}{I} \cdot \frac{1}{3} \cdot \frac{I}{I + J + K} \cdot (1 - B_{Q3}) \)
  – \( B_j = \frac{1}{J} \cdot \frac{1}{3} \cdot \left[ \frac{J}{I + J + K} + \frac{J}{J + K} \right] \cdot (1 - B_{Q3}) \)
  – \( B_k = \frac{1}{K} \cdot \frac{1}{3} \cdot \left[ \frac{K}{I + J + K} + \frac{K}{J + K} + 1 \right] \cdot (1 - B_{Q3}) \)
• in the specific case (when there is no request \( k \) with a priority 2 (\( K = 0 \))):
  – \( B_i = \frac{1}{I} \cdot \frac{1}{2} \cdot \frac{I}{I + J} \cdot (1 - B_{Q3}) \)
  – \( B_j = \frac{1}{J} \cdot \frac{1}{2} \cdot \left[ \frac{J}{I + J} + 1 \right] \cdot (1 - B_{Q3}) \)

Consequently, the GPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

### 16.4.12 GPDMA triggered transfer

A programmed GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by GPDMA_CxTR2.TRIGPOL[1:0] and GPDMA_CxTR2.TRIGSEL[5:0] (see Section 16.3.7 for the trigger selection).

The triggered transfer, as defined by the trigger mode in GPDMA_CxTR2.TRIMG[1:0], can be at LLI data transfer level, to condition the first burst read of a block, the first burst read of a 2D/repeated block for channel \( x \) (\( x = 6 \) to 7), or each programmed single read. The trigger mode can also be programmed to condition the LLI link transfer (see TRIMG[1:0] in GPDMA_CxTR2 for more details).
**Trigger hit memorization and trigger overrun flag generation**

The GPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[5:0] are not modified, and the channel is enabled.

Transferring a next LLIn+1, that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLIn trigger.

**Caution:** After a first new trigger hitn+1 is memorized, if another trigger hitn+2 is detected and if the hitn triggered transfer is still not completed, hitn+2 is lost and not memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

The figure below illustrates the trigger hit, memorization and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

**Figure 84. Trigger hit, memorization and overrun waveform**

![Trigger hit, memorization and overrun waveform](image)

**Note:** The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

**16.4.13 GPDMA circular buffering with linked-list programming**

**GPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers, with a linear addressing channel**

For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half transfer and complete transfer.
events/interrupts generation (GPDMA_CxCR.HTIE = 1 and GPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI0 is configured for the first block transfer with the linear addressing channel. A continuously-executed LLI1 is needed to restore the memory source (or destination) start address, for the memory-to-peripheral transfer (respectively the peripheral-to-memory transfer). GPDMA automatically reloads the initially programmed GPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore GPDMA_CxBR1.

The figure below illustrates this programming with a linear addressing GPDMA channel and a source circular buffer.

**Figure 85. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel**

![Diagram of GPDMA circular buffer programming](image)

**Note:** With a 2D addressing channel, the user may use a single LLI with GPDMA_CxBR1.BRC[10:0] = 1, and program a negative memory block address offset with GDMA_CxBR2 and GDMA_CxBR1, in order to jump back to the memory source or the destination start address.

If circular buffering must be executed after some other transfers over the shared GPDMA channel x, the before-last LLI_{N-1} in memory is needed to configure the first block transfer. And the last LLI_N restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory transfer).
The figure below illustrates this programming with a linear addressing shared GPDMA channel, and a source circular buffer.

Figure 86. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel

16.4.14 GPDMA transfer in peripheral flow-control mode

A peripheral with the peripheral flow-control mode feature can decide to early terminate a GPDMA block transfer, provided that the allocated channel is implemented with this feature (see Section 16.3.6).

If the related GPDMA channel x is also programmed in peripheral flow-control mode (GPDMA_CxTR2.PFREQ = 1):

- The GPDMA block transfer starts as follows:
  - If GPDMA_CxBR1.BNDT[15:0] ≠ 0, the programmed value is internally taken into account by the GPDMA hardware.
  - If GPDMA_CxBR1.BNDT[15:0] = 0, the GPDMA hardware internally considers a 64-Kbyte value for the maximum source block size to be transferred.

- The GPDMA block transfer is completed as soon as the first occurrence of one of the following condition occurs:
  - when GPDMA_CxBR1.BNDT[15:0] = 0
  - when the peripheral early terminates the block. The complete transfer event is generated if programmed, depending on GPDMA_CxTR2 (see GPDMA channel x transfer register 2 (GPDMA_CxTR2)). Then the software can read the current number of transferred bytes from the source (GPDMA_CxBR1.BNDT[15:0]), and/or read the current source or destination address of the buffer in memory (GPDMA_CxSAR[31:0] or GPDMA_CxDAR[31:0]).
In peripheral flow-control mode:
- a destination peripheral with a hardware requested transfer is not supported:
  memory-to-peripheral transfer is not supported.
- Data packing from a source peripheral is not supported.
- 2D/repeated block is not supported.
- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

16.4.15 GPDMA secure/nonsecure channel

The GPDMA controller is compliant with the TrustZone hardware architecture at channel level, partitioning all its resources so that they exist in one of the secure and nonsecure worlds at any given time.

Any channel $x$ is a secure or a nonsecure hardware resource, as configured by GPDMA_SECCFGGR.SEC$x$.

When a channel $x$ is configured in secure state by a secure and privileged agent, the following access control rules are applied:
- A nonsecure read access to a register field of this channel is forced to return 0, except for GPDMA_SECCFGGR, GPDMA_PRIVCFGGR and GPDMA_RCFGLOCKR that are readable by a nonsecure agent.
- A nonsecure write access to a register field of this channel has no impact.

When a channel $x$ is configured in secure state, a secure agent can configure separately as secure or nonsecure the GPDMA data transfer from the source (GPDMA_CxTR1.SSEC) and the GPDMA data transfer to the destination (GPDMA_CxTR1.DSEC).

When a channel $x$ is configured in secure state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with secure transfers via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a secure bus that reflects GPDMA_SECCFGGR, to keep the other peripherals informed of the secure/nonsecure state of each GPDMA channel $x$.

The GPDMA also generates a security illegal access pulse signal on an illegal nonsecure access to a secure GPDMA register. This signal is routed to the TrustZone interrupt controller.

When the secure software must switch a channel from a secure state to a nonsecure state, the secure software must abort the channel or wait until the secure channel is completed before switching. This is needed to dynamically re-allocate a channel to a next nonsecure transfer as a nonsecure software is not allowed to do so and must have GPDMA_CxCR.EN = 0 before the nonsecure software can reprogram the GPDMA_CxCR for a next transfer. The secure software may reset not only the channel $x$ (GPDMA_CxCR.RESET = 1) but also the full channel $x$ register file to its reset value.
16.4.16 GPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via GPDMA_PRIVCFGx.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:

- An unprivileged read access to a register field of this channel is forced to return 0, except for GPDMA_PRIVCFGx, GPDMA_SECCEFGR and GPDMA_RCFGLOCKR that are readable by an unprivileged agent.
- An unprivileged write access to a register field of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a privileged bus that reflects GPDMA_PRIVCFGx, to keep the other peripherals informed of the privileged/unprivileged state of each GPDMA channel x.

When the privileged software must switch a channel from a privileged state to an unprivileged state, the privileged software must abort the channel or wait until that the privileged channel is completed before switching. This is needed to dynamically re-allocate a channel to a next unprivileged transfer as an unprivileged software is not allowed to do so, and must have GPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the GPDMA_CxCR for a next transfer. The privileged software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

16.4.17 GPDMA error management

The GPDMA is able to manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

on a bus access (as a AHB single or a burst) to the source or the destination

- The source or destination target reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware). The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the data error (GPDMA_CxSR.DTEF = 1).
- After a GPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After a GPDMA data transfer error, the user must issue a channel reset (set GPDMA_CxCR.RESET) to reset the hardware GPDMA channel data path and the content of the FIFO, before the user enables again the same channel for a next transfer.
Link transfer error

on a tentative update of a GPDMA channel register from the programmed LLI in the memory

- The linked-list memory reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware), the channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the link error (GPDMA_CxSR.ULEF = 1).
- After a GPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (GPDMA_CxLBAR and GPDMA_CxLLR), including the protection attributes.
- After a GPDMA link error, the user must explicitly write the linked-list register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2), before the user enables again the same channel for a next transfer.

User setting error

on a tentative execution of a GPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (GPDMA_CxCR.EN forced and cleared by the GPDMA hardware) preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and a user setting error (GPDMA_CxSR.USEF = 1).
- After a GPDMA user setting error, the user must perform a debug session, taking care of the GPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxLLR.UB1 = 0)
  - a programmed non-null source block size being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR1.BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - when in packing/unpacking mode (if PAM[1] = 1), a programmed non-null source block size being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR1.BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - a programmed unaligned source start address, being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxSAR[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 6 to 7): a programmed unaligned source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxTR3.SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 6 to 7): a programmed unaligned block repeated source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR2.BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxDAR[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - for channel x (x = 6 to 7): a programmed unaligned destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxTR3.DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
16.4.18 **GPDMA autonomous mode**

To save dynamic power consumption while the GPDMA executes the programmed linked-list transfers, the GPDMA hardware automatically manages its own clock gating and generates a clock request output signal to the RCC, whenever the device is in Run or low-power modes, provided that the RCC is programmed with the corresponding GPDMA enable control bits.

For more details about the RCC programming, refer to the RCC section of the reference manual.

For mode details about the availability of the GPDMA autonomous feature vs the device low-power modes, refer to Section 16.3.3.

The user can program and schedule the execution of a given GPDMA transfer at a LLI<sub>n</sub> level of a GPDMA channel <i>x</i>, with GPDMA_CxTR2 as follows:

- The software controls and conditions the input of a transfer with TRIGM[1:0], TRIGPOL[1:0], TRIGSEL[5:0], SWREQ and REQSEL[7:0] for the input trigger and request.

- The software controls and signals the output of a transfer with TCEM[1:0] for generating or not a transfer complete event, and generating or not an associated half data transfer event).

See GPDMA channel <i>x</i> transfer register 2 (GPDMA_CxTR2) for more details.

When used in low-power modes, this functionality enables a CPU wake-up on a specific transfer completion by the enabled GPDMA transfer complete interrupt (GPDMA_CxCR.TCIE = 1) or/and enables to continue with the autonomous GPDMA for operating another LLI<sub>n+1</sub> transfer over the same channel.

The output channel <i>x</i> transfer complete event, gpdma_chx_tc, can be programmed as a selected input trigger for a channel if this event is looped-back and connected at the GPDMA level (see Section 16.3.7), allowing autonomous and fine GPDMA inter-channel transfer scheduling, without needing a cleared transfer complete flag (TCF).

A given GPDMA channel <i>x</i> asserts its clock request in one of the following conditions:

- if the next transfer to be executed is programmed as conditioned by a trigger (GPDMA_CxTR2.TRIGPOL[1:0] and GPDMA_CxTR2.TRIGM[1:0]), only when the trigger hit occurs.

- if the next transfer to be executed is not conditioned by a trigger:
  - if GPDMA_CxTR2.SWREQ = 0, only when the hardware request is asserted by the selected peripheral.
The GPDMA channel \(x\) releases its clock request as soon as all the following conditions are met:

- The transfer to be executed is completed.
- The GPDMA channel \(x\) is not immediately ready and requested to execute the next transfer.
- If a channel \(x\) interrupt was raised, all the flags of the status register that can cause this interrupt, are cleared by a software agent.

When one channel asserts its clock request, the GPDMA asserts its clock request to the RCC. When none channel asserts its clock request, the GPDMA releases its clock request to the RCC.

### 16.5 GPDMA in debug mode

When the microcontroller enters debug mode (core halted), any channel \(x\) can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

**Note:** In debug mode, GPDMA\_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, GPDMA\_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

### 16.6 GPDMA in low-power modes

#### Table 132. Effect of low-power modes on GPDMA

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. GPDMA interrupts cause the device to exit Sleep mode.</td>
</tr>
<tr>
<td>Stop(^{(1)})</td>
<td>The content of the GPDMA registers is kept when entering Stop mode. The content of the GPDMA registers can be autonomously updated by a next linked-list item from memory, to perform autonomous data transfers. GPDMA interrupts can cause the device to exit Stop mode(^{(1)}).</td>
</tr>
<tr>
<td>Standby</td>
<td>The GPDMA is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

---

\(^{(1)}\) Refer to [Section 16.3.3](#) to know if any Stop mode is supported.
16.7 GPDMA interrupts

There is one GPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 133. GPDMA interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Interrupt enable</th>
<th>Event flag</th>
<th>Event clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPDMA_Chx</td>
<td>Transfer complete</td>
<td>GPDMA_CxCR.TCIE</td>
<td>GPDMA_CxSR.TCF</td>
<td>Write 1 to GPDMA_CxFCR.TCF</td>
</tr>
<tr>
<td></td>
<td>Half transfer</td>
<td>GPDMA_CxCR.HTIE</td>
<td>GPDMA_CxSR.HTF</td>
<td>Write 1 to GPDMA_CxFCR.HTF</td>
</tr>
<tr>
<td></td>
<td>Data transfer error</td>
<td>GPDMA_CxCR.DTEIE</td>
<td>GPDMA_CxSR.DTEF</td>
<td>Write 1 to GPDMA_CxFCR.DTEF</td>
</tr>
<tr>
<td></td>
<td>Update link error</td>
<td>GPDMA_CxCR.ULEIE</td>
<td>GPDMA_CxSR.ULEF</td>
<td>Write 1 to GPDMA_CxFCR.ULEF</td>
</tr>
<tr>
<td></td>
<td>User setting error</td>
<td>GPDMA_CxCR.USEIE</td>
<td>GPDMA_CxSR.USEF</td>
<td>Write 1 to GPDMA_CxFCR.USEF</td>
</tr>
<tr>
<td></td>
<td>Suspended</td>
<td>GPDMA_CxCR.SUSPIE</td>
<td>GPDMA_CxSR.SUSPF</td>
<td>Write 1 to GPDMA_CxFCR.SUSPF</td>
</tr>
<tr>
<td></td>
<td>Trigger overrun</td>
<td>GPDMA_CxCR.TOFIE</td>
<td>GPDMA_CxSR.TOF</td>
<td>Write 1 to GPDMA_CxFCR.TOF</td>
</tr>
</tbody>
</table>

A GPDMA channel x event may be:
- a transfer complete
- a half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

Note: When a channel x transfer complete event occurs, the output signal gpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:
- the corresponding interrupt event xx is enabled (GPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (GPDMA_CxSR.xx = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into GPDMA_CxFCR.xx).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by GPDMA_CxTR2.TCEM[1:0] as follows:
- A transfer complete event is a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].
A half transfer event is an half block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

A half-block transfer occurs when half of the source block size bytes (rounded-up integer of GPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded-up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2) is transferred to the destination.

See GPDMA channel x transfer register 2 (GPDMA_CxTR2) for more details.

A transfer error rises in one of the following situations:

- during a single/burst data transfer from the source or to the destination (DTEF)
- during an update of a GPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of a GPDMA channel with an unauthorized setting (USEF)

The user must perform a debug session to correct the GPDMA channel programming versus the USEF root causes list (see Section 16.4.17).

A trigger overrun is described in Trigger hit memorization and trigger overrun flag generation.

16.8 GPDMA registers

The GPDMA registers must be accessed with an aligned 32-bit word data access.

16.8.1 GPDMA secure configuration register (GPDMA_SECCFGR)

Address offset: 0x00
Reset value: 0x0000 0000

A write access to this register must be secure and privileged. A read access is secure or nonsecure, privileged or unprivileged.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (when GPDMA_CxCR.EN = 0), to securely allocate individually any channel x to the secure or nonsecure world.

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

Bits 31:8 Reserved, must be kept at reset value.
16.8.2 GPDMA privileged configuration register (GPDMA_PRIVCFGR)

Address offset: 0x04
Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged, secure or nonsecure.

This register can mix secure and nonsecure information. If a channel x is configured as secure (GPDMA_SECCFGR.SECx = 1), the PRIVx bit can be written only by a secure (and privileged) agent.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (GPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

Bits 7:0  \textbf{SECx}: secure state of channel x (x = 7 to 0)

0: nonsecure
1: secure

16.8.3 GPDMA configuration lock register (GPDMA_RCFGLOCKR)

Address offset: 0x08
Reset value: 0x0000 0000

This register can be written by a software agent with secure privileged attributes in order to individually lock, for example at boot time, the secure privileged attributes of any GPDMA...
channel/resource (to lock the setting of GPDMA_CxSECCFGR and GPDMA_CxPRIVCFGR for any channel x, for example at boot time).

A read access may be privileged or unprivileged, secure or nonsecure.

**Note:** If TZEN = 0, this register cannot be written.

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

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

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

Bits 7:0 **LOCKx**: lock the configuration of GPDMA_SECCFGR.SECx and GPDMA_PRIVCFGR.PRIVx, until a global GPDMA reset (x = 7 to 0).

This bit is cleared after reset and, once set, it cannot be reset until a global GPDMA reset.

0: secure privilege configuration of the channel x is writable.

1: secure privilege configuration of the channel x is not writable.

### 16.8.4 GPDMA nonsecure masked interrupt status register (GPDMA_MISR)

Address offset: 0x0C

Reset value: 0x0000 0000

This register is a read register.

This is a nonsecure register, containing the masked interrupt status bit MISx for each nonsecure channel x (channel x configured with GPDMA_SECCFGR.SECx = 0). It is a logical OR of all the flags of GPDMA_CxSR, each source flag being enabled by the corresponding interrupt enable bit of GPDMA_CxCR.

Every bit is deasserted by hardware when writing 1 to the corresponding flag clear bit in GPDMA_CxFCR.

If a channel x is in secure state (GPDMA_SECCFGR.SECx = 1), a read access to the masked interrupt status bit MISx of this channel x returns zero.

This register may mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full nonsecure interrupt status. An unprivileged software is restricted to read the status of unprivileged (and nonsecure) channels, other privileged bit fields returning zero.

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

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

Bits 31:8 Reserved, must be kept at reset value.
16.8.5 **GPDMA secure masked interrupt status register (GPDMA_SMISR)**

Address offset: 0x10  
Reset value: 0x0000 0000

This is a secure read register, containing the masked interrupt status bit MISx for each secure channel x (GPDMA_SECCFGR.SECx = 1). It is a logical OR of all the GPDMA_CxSR flags, each source flag being enabled by the corresponding GPDMA_CxCR interrupt enable bit.

Every bit is deasserted by hardware when securely writing 1 to the corresponding GPDMA_CxFCR flag clear bit.

This register does not contain any information about a nonsecure channel.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full secure interrupt status. An unprivileged software is restricted to read the status of unprivileged and secure channels, other privileged bit fields returning zero.

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

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

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

Bits 7:0  **MISx**: masked interrupt status of the secure channel x (x = 7 to 0)  
0: no interrupt occurred on the secure channel x  
1: an interrupt occurred on the secure channel x
16.8.6  **GPDMA channel x linked-list base address register (GPDMA_CxLBAR)**

Address offset: 0x50 + 0x80 * x (x = 0 to 7)
Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x GPDMA_PRIVCFGR.PRIVx.

This register is either secure or nonsecure depending on the secure state of the channel x (GPDMA_SECCFGR.SECx).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the GPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit GPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

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

Bits 31:16 **LBA[31:16]**: linked-list base address of GPDMA channel x

Bits 15:0  Reserved, must be kept at reset value.

16.8.7  **GPDMA channel x flag clear register (GPDMA_CxFCR)**

Address offset: 0x5C + 0x80 * x (x = 0 to 7)
Reset value: 0x0000 0000

This is a write register, secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx) and privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

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

Bits 31:15  Reserved, must be kept at reset value.
Bit 14 **TOF**: trigger overrun flag clear  
0: no effect  
1: corresponding TOF flag cleared

Bit 13 **SUSPF**: completed suspension flag clear  
0: no effect  
1: corresponding SUSPF flag cleared

Bit 12 **USEF**: user setting error flag clear  
0: no effect  
1: corresponding USEF flag cleared

Bit 11 **ULEF**: update link transfer error flag clear  
0: no effect  
1: corresponding ULEF flag cleared

Bit 10 **DTEF**: data transfer error flag clear  
0: no effect  
1: corresponding DTEF flag cleared

Bit 9 **HTF**: half transfer flag clear  
0: no effect  
1: corresponding HTF flag cleared

Bit 8 **TCF**: transfer complete flag clear  
0: no effect  
1: corresponding TCF flag cleared

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

### 16.8.8 GPDMA channel x status register (GPDMA_CxSR)

Address offset: $0x60 + 0x80 \times x$ (x = 0 to 7)  
Reset value: $0x0000\ 0001$

This is a read register, reporting the channel status.

This register is secure or nonsecure, depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of the channel (GPDMA_PRIVCFGPR.PRIVx).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res.</td>
<td>TOF</td>
<td>SUSPF</td>
<td>USEF</td>
<td>ULEF</td>
<td>DTEF</td>
<td>HTF</td>
<td>TCF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>IDLEF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.
Bits 23:16  **FIFOL[7:0]:** monitored FIFO level
Number of available write beats in the FIFO, in units of the programmed destination data width (see **GPDMA_CxTR1.DDW_LOG2[1:0]**, in units of bytes, half-words, or words).

*Note:* After having suspended an active transfer, the user may need to read **FIFOL[7:0]**, additionally to **GPDMA_CxBR1.BDNT[15:0]** and **GPDMA_CxBR1.BRC[10:0]**, to know how many data have been transferred to the destination. Before reading, the user may wait for the transfer to be suspended (**GPDMA_CxSR.SUSPF = 1**).

Bit 15  Reserved, must be kept at reset value.

Bit 14  **TOF:** trigger overrun flag
0: no trigger overrun event
1: a trigger overrun event occurred

Bit 13  **SUSPF:** completed suspension flag
0: no completed suspension event
1: a completed suspension event occurred

Bit 12  **USEF:** user setting error flag
0: no user setting error event
1: a user setting error event occurred

Bit 11  **ULEF:** update link transfer error flag
0: no update link transfer error event
1: a master bus error event occurred while updating a linked-list register from memory

Bit 10  **DTEF:** data transfer error flag
0: no data transfer error event
1: a master bus error event occurred on a data transfer

Bit 9  **HTF:** half transfer flag
0: no half transfer event
1: a half transfer event occurred

A half transfer event is either a half block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode (**GPDMA_CxTR2.TCEM[1:0]**).

A half block transfer occurs when half of the bytes of the source block size (rounded up integer of **GPDMA_CxBR1.BNDT[15:0]/2**) has been transferred to the destination.

A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded up integer of (**GPDMA_CxBR1.BRC[10:0] + 1) / 2)) has been transferred to the destination.

Bit 8  **TCF:** transfer complete flag
0: no transfer complete event
1: a transfer complete event occurred

A transfer complete event is either a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode (**GPDMA_CxTR2.TCEM[1:0]**).

Bits 7:1  Reserved, must be kept at reset value.

Bit 0  **IDLEF:** idle flag
0: channel not in idle state
1: channel in idle state

This idle flag is deasserted by hardware when the channel is enabled (**GPDMA_CxCR.EN = 1**) with a valid channel configuration (no **USEF** to be immediately reported).

This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).
### 16.8.9 GPDMA channel x control register (GPDMA_CxCR)

Address offset: 0x64 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

This register is used to control a channel (activate, suspend, abort or disable it).

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 23:22</td>
<td><strong>PRIO[1:0]</strong>: priority level of the channel x GPDMA transfer versus others</td>
</tr>
<tr>
<td>00</td>
<td>low priority, low weight</td>
</tr>
<tr>
<td>01</td>
<td>low priority, mid weight</td>
</tr>
<tr>
<td>10</td>
<td>low priority, high weight</td>
</tr>
<tr>
<td>11</td>
<td>high priority</td>
</tr>
</tbody>
</table>

*Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.*

<table>
<thead>
<tr>
<th>Bit 21:18</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 17</td>
<td><strong>LAP</strong>: linked-list allocated port</td>
</tr>
<tr>
<td>0</td>
<td>port 0 (AHB) allocated</td>
</tr>
<tr>
<td>1</td>
<td>port 1 (AHB) allocated</td>
</tr>
</tbody>
</table>

*Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.*

<table>
<thead>
<tr>
<th>Bit 16</th>
<th><strong>LSM</strong>: Link step mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>channel executed for the full linked-list and completed at the end of the last LLI (GPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0 and UT3 = UB2 = 0). Then GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0.</td>
</tr>
<tr>
<td>1</td>
<td>channel executed <strong>once</strong> for the current LLI</td>
</tr>
<tr>
<td>First the (possible 1D/repeated) block transfer is executed as defined by the current internal register file until GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Secondly the next linked-list data structure is conditionally uploaded from memory as defined by GPDMA_CxLLR. Then channel execution is completed.</td>
<td></td>
</tr>
</tbody>
</table>

*Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.*

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 14</td>
<td><strong>TOIE</strong>: trigger overrun interrupt enable</td>
</tr>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>
Bit 13 **SUSPIE**: completed suspension interrupt enable
   0: interrupt disabled
   1: interrupt enabled

Bit 12 **USEIE**: user setting error interrupt enable
   0: interrupt disabled
   1: interrupt enabled

Bit 11 **ULEIE**: update link transfer error interrupt enable
   0: interrupt disabled
   1: interrupt enabled

Bit 10 **DTEIE**: data transfer error interrupt enable
   0: interrupt disabled
   1: interrupt enabled

Bit 9 **HTIE**: half transfer complete interrupt enable
   0: interrupt disabled
   1: interrupt enabled

Bit 8 **TCIE**: transfer complete interrupt enable
   0: interrupt disabled
   1: interrupt enabled

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

Bit 2 **SUSP**: suspend
   Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 2. Else:
   Software must write 1 in order to suspend an active channel (channel with an ongoing GPDMA transfer over its master ports).
   The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in Figure 68.
   0: write: resume channel, read: channel not suspended
   1: write: suspend channel, read: channel suspended.

Bit 1 **RESET**: reset
   This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the FIFO, the channel internal state, SUSP and EN bits (whatever is written receptively in bit 2 and bit 0).
   The reset is effective when the channel is in steady state, meaning one of the following:
   - active channel in suspended state (GPDMA_CxSR.SUSPF = 1 and GPDMA_CxSR.IDLEF = GPDMA_CxCR.EN = 1)
   - channel in disabled state (GPDMA_CxSR.IDLEF = 1 and GPDMA_CxCR.EN = 0).
   After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR) before enabling again the channel (see the programming sequence in Figure 69).
   0: no channel reset
   1: channel reset
Bit 0 **EN**: enable
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 0. Else:
this bit is deasserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example if LSM = 1 at the end of a single execution of the LLI).
Else, this bit can be asserted by software.
Writing 0 into this EN bit is ignored.
0: write: ignored, read: channel disabled
1: write: enable channel, read: channel enabled

16.8.10 **GPDMA channel x transfer register 1 (GPDMA_CxTR1)**

Address offset: 0x90 + 0x80 * x (x = 0 to 7)
Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFG.SECx) except for secure DSEC and SSEC, privileged or non-privileged, depending on the privileged state of the channel x in GPDMA_PRIVCFG.PRIVx.

This register controls the transfer of a channel x.
This register must be written when GPDMA_CxCR.EN = 0.
This register is read-only when GPDMA_CxCR.EN = 1.
This register must be written when the channel is completed. Then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.
In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UT1 = 1.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSEC</td>
<td>DAP</td>
<td>...</td>
<td>...</td>
<td>DHX</td>
<td>DBX</td>
<td>DBL[5:0]</td>
<td>DINC</td>
<td>...</td>
<td>DDW_LOG2[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSEC</td>
<td>SAP</td>
<td>SBX</td>
<td>PAM[1:0]</td>
<td>...</td>
<td>SBL[5:0]</td>
<td>SINC</td>
<td>...</td>
<td>SDW_LOG2[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **DSEC**: security attribute of the GPDMA transfer to the destination
If GPDMA_SECCFG.SECx = 1 and the access is secure:
0: GPDMA transfer nonsecure
1: GPDMA transfer secure
This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when GPDMA_SECCFG.SECx = 1. A secure write is ignored when GPDMA_SECCFG.SECx = 0.
When GPDMA_SECCFG.SECx is deasserted, this DSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the GPDMA transfer to the destination is nonsecure.
Bit 30  **DAP**: destination allocated port
This bit is used to allocate the master port for the destination transfer
0: port 0 (AHB) allocated
1: port 1 (AHB) allocated

*Note:* This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 29:28  Reserved, must be kept at reset value.

Bit 27  **DHX**: destination half-word exchange
If the destination data size is shorter than a word, this bit is ignored.
If the destination data size is a word:
0: no halfword-based exchanged within word
1: the two consecutive (post PAM) half-words are exchanged in each destination word.

Bit 26  **DBX**: destination byte exchange
If the destination data size is a byte, this bit is ignored.
If the destination data size is not a byte:
0: no byte-based exchange within half-word
1: the two consecutive (post PAM) bytes are exchanged in each destination half-word.

Bits 25:20  **DBL_1[5:0]**: destination burst length minus 1, between 0 and 63
The burst length unit is one data named beat within a burst. If DBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width **DDW_LOG2[1:0]**.

*Note:* If a burst transfer crossed a 1-Byte address boundary on an AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 19  **DINC**: destination incrementing burst
0: fixed burst
1: contiguously incremented burst
The destination address, pointed by GPDMA_CxDAR, is kept constant after a burst beat/single transfer, or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 18  Reserved, must be kept at reset value.

Bits 17:16  **DDW_LOG2[1:0]**: binary logarithm of the destination data width of a burst, in bytes
00: byte
01: half-word (2 bytes)
10: word (4 bytes)
11: user setting error reported and no transfer issued

*Note:* Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.

A destination burst transfer must have an aligned address with its data width (start address GPDMA_CxDAR[2:0] and address offset GPDMA_CxTR3.DAO[2:0], versus **DDW_LOG2[1:0]**). Otherwise a user setting error is reported and no transfer is issued.
Bit 15  **SSEC**: security attribute of the GPDMA transfer from the source
   If GPDMA_SECCFGR.SECx = 1 and the access is secure:
   0: GPDMA transfer nonsecure
   1: GPDMA transfer secure
   This is a secure register bit. This bit can only be read by a secure software. This bit must be
   written by a secure software when GPDMA_SECCFGR.SECx = 1. A secure write is ignored
   when GPDMA_SECCFGR.SECx = 0.
   When GPDMA_SECCFGR.SECx is deasserted, this SSEC bit is also deasserted by
   hardware (on a secure reconfiguration of the channel as nonsecure), and the GPDMA
   transfer from the source is nonsecure.

Bit 14  **SAP**: source allocated port
   This bit is used to allocate the master port for the source transfer
   0: port 0 (AHB) allocated
   1: port 1 (AHB) allocated
   *Note:* This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 13  **SBX**: source byte exchange within the unaligned half-word of each source word
   If the source data width is shorter than a word, this bit is ignored.
   If the source data width is a word:
   0: no byte-based exchange within the unaligned half-word of each source word
   1: the two consecutive bytes within the unaligned half-word of each source word are
      exchanged.

Bits 12:11  **PAM[1:0]**: padding/alignment mode
   If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a burst destination transfer is
   equal to the data width of a burst source transfer, these bits are ignored.
   Else, in the following enumerated values, the condition PAM_1 is when destination data width
   is higher than source data width, and the condition PAM_2 is when source data width is
   higher than destination data width.
   Condition: PAM_1
   00: source data is transferred as right aligned, padded with 0s up to the destination data
      width
   01: source data is transferred as right aligned, sign extended up to the destination data width
   10-11: successive source data are FIFO queued and packed at the destination data width, in
   a left (LSB) to right (MSB) order (named little endian), before a destination transfer
   Condition: PAM_2
   00: source data is transferred as right aligned, left-truncated down to the destination data
      width
   01: source data is transferred as left-aligned, right-truncated down to the destination data
      width
   10-11: source data is FIFO queued and unpacked at the destination data width, to be
   transferred in a left (LSB) to right (MSB) order (named little endian) to the destination
   *Note:* If the transfer from the source peripheral is configured with peripheral flow-control mode
   (SWREQ = 0 and PFREQ = 1 and DREQ = 0), and if the destination data width > the
   source data width, packing is not supported.

Bit 10  **Reserved**, must be kept at reset value.
Bits 9:4  **SBL_1[5:0]**: source burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If SBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width SDW_LOG2[1:0].

**Note:** If a burst transfer crossed a 1-Byte address boundary on an AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 3  **SINC**: source incrementing burst

0: fixed burst

1: contiguously incremented burst

The source address, pointed by GPDMA_CxSAR, is kept constant after a burst beat/single transfer or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 2  Reserved, must be kept at reset value.

Bits 1:0  **SDW_LOG2[1:0]**: binary logarithm of the source data width of a burst in bytes

00: byte

01: half-word (2 bytes)

10: word (4 bytes)

11: user setting error reported and no transfer issued

**Note:** Setting a 8-byte data width causes a user setting error to be reported and no transfer is issued.

A source block size must be a multiple of the source data width (GPDMA_CxBR1.BNDT[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and no transfer is issued.

A source burst transfer must have an aligned address with its data width (start address GPDMA_CxSAR[2:0] versus SDW_Log2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.
### 16.8.11 GPDMA channel x transfer register 2 (GPDMA_CxTR2)

Address offset: 0x94 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGPR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory, if GPDMA_CxLLR.UT2 = 1.

<table>
<thead>
<tr>
<th>Bit 31-30</th>
<th>Description 1</th>
<th>Description 2</th>
<th>Description 3</th>
<th>Description 4</th>
<th>Description 5</th>
<th>Description 6</th>
<th>Description 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCEM[1:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>TRIGPOL[1:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>TRIGSEL[5:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>Bit 15-14</td>
<td>Description 8</td>
<td>Description 9</td>
<td>Description 10</td>
<td>Description 11</td>
<td>Description 12</td>
<td>Description 13</td>
<td>Description 14</td>
</tr>
<tr>
<td>TRIGM[1:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>PFREQ</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>BREQ</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>DREQ</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>SWREQ</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>REQSEL[7:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
**Bits 31:30**  **TCEM[1:0]:** transfer complete event mode

These bits define the transfer granularity for the transfer complete and half transfer complete events generation.

00: at block level (when GPDMA_CxBR1.BNDT[15:0] = 0): the complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

**Note:** If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

01: channel x (x = 0 to 5), same as 00, channel x (x = 6 to 7), at 2D/repeated block level (when GPDMA_CxBR1.BRC[10:0] = 0 and GPDMA_CxBR1.BNDT[15:0] = 0). The complete (and the half) transfer event is generated at the end (respectively half of the end) of the 2D/repeated block.

**Note:** If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

10: at LLI level: the complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half transfer event is generated at the half of the LLI data transfer. The LLI data transfer is a block transfer or a 2D/repeated block transfer for channel x (x = 6 to 7), if any data transfer.

**Note:** If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then the half transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI.

11: at channel level: the complete transfer event is generated at the end of the last LLI transfer. The half transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address GPDMA_CxLLR.LA[15:2] to zero and clears all the GPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UT3 and UB2). If the channel transfer is continuous/infinite, no event is generated.

**Bits 29:26**  **Reserved, must be kept at reset value.**

**Bits 25:24**  **TRIGPOL[1:0]:** trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[5:0].

00: no trigger (masked trigger event)

01: trigger on the rising edge

10: trigger on the falling edge

11: same as 00

**Bits 23:22**  **Reserved, must be kept at reset value.**

**Bits 21:16**  **TRIGSEL[5:0]:** trigger event input selection

These bits select the trigger event input of the GPDMA transfer (as per *Section 16.3.7*), with an active trigger event if TRIGPOL[1:0] ≠ 00.
Bits 15:14 **TRIGM[1:0]: trigger mode**

These bits define the transfer granularity for its conditioning by the trigger.

- If the channel x is enabled (GPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 00 or 11, these TRIGM[1:0] bits are ignored.
- Else, a GPDMA transfer is conditioned by at least one trigger hit:
  - 00: at block level: the first burst read of each block transfer is conditioned by one hit trigger (channel x = 12 to 15, for each block if a 2D/repeated block is configured with GPDMA_CxBR1.BRC[10:0] ≠ 0).
  - 01: channel x (x = 0 to 5), same as 00; channel x (x = 6 to 7), at 2D/repeated block level. The first burst read of a 2D/repeated block transfer is conditioned by one hit trigger.
  - 10: at link level: a LLI link transfer is conditioned by one hit trigger. The LLI data transfer (if any) is not conditioned.
  - 11: at programmed burst level: If SWREQ = 1, each programmed burst read is conditioned by one hit trigger. If SWREQ = 0, each programmed burst that is requested by the selected peripheral, is conditioned by one hit trigger.

- If the peripheral is programmed as a source (DREQ = 0) of the LLI data transfer, each programmed burst read is conditioned.
- If the peripheral is programmed as a destination (DREQ = 1) of the LLI data transfer, each programmed burst is conditioned. The first memory burst read of a (possibly 2D/repeated) block, also named as the first ready FIFO-based source burst, is gated by the occurrence of both the hardware request and the first trigger hit.

The GPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 01 or respectively TRIGPOL[1:0] = 10).

The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[5:0] is not modified, and the channel is enabled.

Transferring a next LLI\(n+1\) that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI\(n\) trigger.

After a first new trigger hit\(n+1\) is memorized, if another second trigger hit\(n+2\) is detected and if the hit\(n\) triggered transfer is still not completed, hit\(n+2\) is lost and not memorized.

- A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1), and an interrupt is generated if enabled (GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

**Note:** When the source block size is not a multiple of the source burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shortened to match the block size. In this case, if TRIGM[1:0] = 11 and (SWREQ = 1 or (SWREQ = 0 and DREQ = 0)), the shortened burst transfer (by singles or/and by bursts of lower length) is conditioned once by the trigger.

When the programmed destination burst is internally shortened by singles or/and by bursts of lower length (versus FIFO size, versus block size, 1-Kbyte boundary address crossing): if the trigger is conditioning the programmed destination burst (if TRIGM[1:0] = 11 and SWREQ = 0 and DREQ = 1), this shortened destination burst transfer is conditioned once by the trigger.

Bit 13 Reserved, must be kept at reset value.
Bit 12  **PFREQ**: Hardware request in peripheral flow control mode

*Important: If a given channel x is not implemented with this feature, this bit is reserved and PFREQ is not present (see Section 16.3.2 for the list of the implemented channels with this feature).*

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in GPDMA control mode. The GPDMA is programmed with GPDMA_CxTR1.BNDT[15:0] and this is internally used by the hardware for the block transfer completion.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in peripheral control mode. The GPDMA block transfer can be early completed by the peripheral itself (see Section 16.3.6 for more details).

**Note**: In peripheral flow control mode, there are the following restrictions:
- no 2D/repeated block support (GPDMA_CxBR1.BRC[10:0] must be set to 0)
- the peripheral must be set as the source of the transfer (DREQ = 0).
- data packing to a wider destination width is not supported (if destination width > source data width, GPDMA_CxTR1.PAM[1] must be set to 0).
- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

Bit 11  **BREQ**: Block hardware request

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a burst level.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see Section 16.3.4).

Bit 10  **DREQ**: destination hardware request

This bit is ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else:

0: selected hardware request driven by a source peripheral (request signal taken into account by the GPDMA transfer scheduler over the source/read port)

1: selected hardware request driven by a destination peripheral (request signal taken into account by the GPDMA transfer scheduler over the destination/write port)

**Note**: If the channel x is activated (GPDMA_CxCR.EN is asserted) with SWREQ = 0 and PFREQ = 1 (peripheral hardware request with peripheral flow-control mode), any software assertion to this DREQ bit is ignored: in peripheral flow-control mode, only a peripheral-to-memory transfer is supported.

Bit 9  **SWREQ**: software request

This bit is internally taken into account when GPDMA_CxCR.EN is asserted.

0: no software request. The selected hardware request REQSEL[7:0] is taken into account.

1: software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[7:0] is ignored.

Bit 8  Reserved, must be kept at reset value.
16.8.12 **GPDMA channel x block register 1 (GPDMA_CxBR1)**

Address offset: 0x98 + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of channel x (GPDMA_PRIVCFGPR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the field BNDT[15:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI
- If GPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

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

Bits 31:16 Reserved, must be kept at reset value.
Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):
- if GPDMA_CxLLR.UB1 = 1, this field is updated by the LLI in the memory.
- if GPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this field is internally restored to the programmed value.
- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] = 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

**Note:** A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (GPDMAM_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

### 16.8.13 GPDMA channel x alternate block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMAM_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of channel x (GPDMAM_PRIVCFGPR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:
- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is
automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0] after each execution of this final LLI

- If GPDMA_CxLLR = 0, BNDT[15:0] and BRC[10:0] are kept as null, channel x is completed.

<table>
<thead>
<tr>
<th>BRDDT C</th>
<th>BRSDT C</th>
<th>DDEC</th>
<th>SDEC</th>
<th>BRC[10:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bit 31 **BRDDEC**: Block repeat destination address decrement

0: at the end of a block transfer, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxBR2.BRDAO to the current GPDMA_CxDAR value (current destination address)

1: at the end of a block transfer, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRDAO from the current GPDMA_CxDAR value (current destination address)

**Note:** On top of this increment/decrement (depending on BRDDEC), GPDMA_CxDAR is in the same time also updated by the increment/decrement (depending on DDEC) of the GPDMA_CxTR3.DAO value, as it is usually done at the end of each programmed burst transfer.

Bit 30 **BRSDEC**: Block repeat source address decrement

0: at the end of a block transfer, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxBR2.BRSAO to the current GPDMA_CxSAR value (current source address)

1: at the end of a block transfer, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRSAO from the current GPDMA_CxSAR value (current source address)

**Note:** On top of this increment/decrement (depending on BRSDEC), GPDMA_CxSAR is in the same time also updated by the increment/decrement (depending on SDEC) of the GPDMA_CxTR3.SAO value, as it is done after any programmed burst transfer.

Bit 29 **DDEC**: destination address decrement

0: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

1: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

Bit 28 **SDEC**: source address decrement

0: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

1: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

Bit 27 **Reserved**, must be kept at reset value.
Bits 26:16 **BRC[10:0]**: Block repeat counter

This field contains the number of repetitions of the current block (0 to 2047). When the channel is enabled, this field becomes read-only. After decrements, this field indicates the remaining number of blocks, excluding the current one. This counter is hardware decremented for each completed block transfer.

Once the last block transfer is completed (BRC[10:0] = BNDT[15:0] = 0):
- If GPDMA_CxLLR.UB1 = 1, all GPDMA_CxBR1 fields are updated by the next LLI in the memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.
- If all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- If GPDMA_CxLLR = 0, this field is kept as zero following the last LLI and data transfer.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):
- If GPDMA_CxLLR.UB1 = 1, this field is updated by the LLI in the memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.
- If all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- If GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

*Note:* A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (GPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.
16.8.14  GPDMA channel x source address register (GPDMA_CxSAR)

Address offset: 0x9C + 0x80 * x (x = 0 to 7)
Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFG.R.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFG.R.PRIVx).

This register configures the source start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.USA = 1.

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

| 31:16 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15:0  |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
Bits 31:0 **SA[31:0]: source address**

This field is the pointer to the address from which the next data is read. During the channel activity, depending on the source addressing mode (GPDMA_CxTR1.SINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.SDW_LOG2[1:0]) after each burst source data, reflecting the next address from which data is read.

During the channel activity, this address is updated after each completed source burst, consequently to:

- the programmed source burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.SINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.SBL_1[5:0] and GPDMA_CxTR1.SDW_LOG2[21:0]

- the additional source incremented/decremented offset value as programmed by GPDMA_CxBR1.SDEC and GPDMA_CxTR3.SAO[12:0].

- once/if completed source block transfer, for a channel x with 2D addressing capability (x = 6 to 7). additional block repeat source incremented/decremented offset value as programmed by GPDMA_CxBR1.BRSDEC and GPDMA_CxBR2.BRSAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.USA = 1.

**Note:** A source address must be aligned with the programmed data width of a source burst (SA[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

When the source block size is not a multiple of the source burst size and is a multiple of the source data width, the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.
16.8.15 GPDMA channel x destination address register (GPDMA_CxDAR)

Address offset: 0xA0 + 0x80 * x (x = 0 to 7)
Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UDA = 1.

<table>
<thead>
<tr>
<th>DA[31:16]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>DA[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 **DA[31:0]: destination address**

This field is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (GPDMA_CxTR1.DINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.DDW_LOG2[21:0]) after each burst destination data, reflecting the next address from which data is written.

During the channel activity, this address is updated after each completed destination burst, consequently to:

- the programmed destination burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.DINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.DBL_1[5:0] and GPDMA_CxTR1.DDW_LOG2[1:0]
- the additional destination incremented/decremented offset value as programmed by GPDMA_CxBR1.DDEC and GPDMA_CxTR3.DAO[12:0].
- once/if completed destination block transfer, for a channel x with 2D addressing capability (x = 6 to 7), the additional block repeat destination incremented/decremented offset value as programmed by GPDMA_CxBR1.BRDEC and GPDMA_CxBR2.BRDAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by the GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.UDA = 1.

**Note:** A destination address must be aligned with the programmed data width of a destination burst [DA[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]]. Else, a user setting error is reported and no transfer is issued.
16.8.16 GPDMA channel x transfer register 3 (GPDMA_CxTR3)

Address offset: 0xA4 + 0x80 * x (x = 6 to 7)
Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGPR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.
This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UT3 = 1.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Access</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:29</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>
| 28:16 | DAO[12:0]: destination address offset increment | rw | 0
| 15:13 | Reserved, must be kept at reset value. | rw | |
| 12:0  | SAO[12:0]: source address offset increment | rw | 0

Note: A destination address offset must be aligned with the programmed data width of a destination burst (DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

Note: A source address offset must be aligned with the programmed data width of a source burst (SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

When the source block size is not a multiple of the destination burst size, and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.
16.8.17 GPDMA channel x block register 2 (GPDMA_CxBR2)

Address offset: 0xA8 + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGPR.PRI{Vx}).

This register controls the transfer of a channel x at a 2D/repeated block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UB2 = 1.

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>BRDAO[15:0]</th>
<th>Block repeated destination address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRDDEC) the current destination address (GPDMA_CxDAR) at the end of a block transfer.</td>
</tr>
<tr>
<td>Note:</td>
<td>A block repeated destination address offset must be aligned with the programmed data width of a destination burst (BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Else a user setting error is reported and no transfer is issued.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BRDAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>BRSAO[15:0]</th>
<th>Block repeated source address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRSDEC) the current source address (GPDMA_CxSAR) at the end of a block transfer.</td>
</tr>
<tr>
<td>Note:</td>
<td>A block repeated source address offset must be aligned with the programmed data width of a source burst (BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Else a user setting error is reported and no transfer is issued.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BRSAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).</td>
<td></td>
</tr>
</tbody>
</table>
16.8.18 GPDMA channel x linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGPR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>UT1</td>
<td>Update GPDMA_CxTR1 from memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: no GPDMA_CxTR1 update</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: GPDMA_CxTR1 update</td>
</tr>
<tr>
<td>30</td>
<td>UT2</td>
<td>Update GPDMA_CxTR2 from memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: no GPDMA_CxTR2 update</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: GPDMA_CxTR2 update</td>
</tr>
<tr>
<td>29</td>
<td>UB1</td>
<td>Update GPDMA_CxBR1 from memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: GPDMA_CxBR1 update</td>
</tr>
<tr>
<td>28</td>
<td>USA</td>
<td>Update GPDMA_CxSAR from memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: no GPDMA_CxSAR update</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: GPDMA_CxSAR update</td>
</tr>
</tbody>
</table>
Bit 27  **UDA**: Update GPDMA\_CxDAR register from memory  
This bit is used to control the update of GPDMA\_CxDAR from the memory during the link transfer.  
0: no GPDMA\_CxDAR update  
1: GPDMA\_CxDAR update  

Bits 26:17  Reserved, must be kept at reset value.  

Bit 16  **ULL**: Update GPDMA\_CxLLR register from memory  
This bit is used to control the update of GPDMA\_CxLLR from the memory during the link transfer.  
0: no GPDMA\_CxLLR update  
1: GPDMA\_CxLLR update  

Bits 15:2  **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure  
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:20] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.  
Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA\_CxTR1, GPDMA\_CxTR2, GPDMA\_CxBR1, GPDMA\_CxSAR, GPDMA\_CxDAR, and GPDMA\_CxLLR).  

*Note*: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.  

Bits 1:0  Reserved, must be kept at reset value.
16.8.19  **GPDMA channel x alternate linked-list address register (GPDMA_CxLLR)**

Address offset: 0xCC + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

```
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>UT1: Update GPDMA_CxTR1 from memory</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>UT2: Update GPDMA_CxTR2 from memory</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>UB1: Update GPDMA_CxBR1 from memory</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>USA: Update GPDMA_CxSAR from memory</td>
<td>rw</td>
</tr>
</tbody>
</table>
```

Bit 31  **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.

0: no GPDMA_CxTR1 update
1: GPDMA_CxTR1 update

Bit 30  **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.

0: no GPDMA_CxTR2 update
1: GPDMA_CxTR2 update

Bit 29  **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer. If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed. GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.

0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: GPDMA_CxBR1 update

Bit 28  **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.

0: no GPDMA_CxSAR update
1: GPDMA_CxSAR update
Bit 27 **UDA**: Update GPDMA_CxDAR register from memory
This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.
0: no GPDMA_CxDAR update
1: GPDMA_CxDAR update

Bit 26 **UT3**: Update GPDMA_CxTR3 from memory
This bit controls the update of GPDMA_CxTR3 from the memory during the link transfer.
0: no GPDMA_CxTR3 update
1: GPDMA_CxTR3 update

Bit 25 **UB2**: Update GPDMA_CxBR2 from memory
This bit controls the update of GPDMA_CxBR2 from the memory during the link transfer.
0: no GPDMA_CxBR2 update
1: GPDMA_CxBR2 update

Bits 24:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory
This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.
0: no GPDMA_CxLLR update
1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:20] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.
Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).

*Note*: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

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

### 16.8.20 GPDMA register map

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | GPDMA_SECCFG |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | GPDMA_PRIVCFG |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | GPDMA_RCFGLOCKR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C   | GPDMA_MISR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Table 134. GPDMA register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10</td>
<td>GPDMAS_MISR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x50+0x80 * x (x=0 to 7)</td>
<td>GPDMACxLBAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LBA[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x54 to 0x58+0x80 * x (x=0 to 7)</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x5C+0x80 * x (x=0 to 7)</td>
<td>GPDMACxCFCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FIFOL[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x60+0x80 * x (x=0 to 7)</td>
<td>GPDMACxCFSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x64+0x80 * x (x=0 to 7)</td>
<td>GPDMACxCFR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x68 to 0x6C+0x80 * x (x=0 to 7)</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x6C+0x80 * x (x=0 to 7)</td>
<td>GPDMACxTR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x94+0x80 * x (x=0 to 7)</td>
<td>GPDMACxTR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TCEM[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x98+0x80 * x (x=0 to 5)</td>
<td>GPDMACxBF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BNDT[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x98+0x80 * x (x=6 to 7)</td>
<td>GPDMACxBF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BRC[10:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x9C+0x80 * x (x=0 to 7)</td>
<td>GPDMACxSAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SA[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xA0+0x80 * x (x=0 to 7)</td>
<td>GPDMACxDAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DA[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xA4+0x80 * x (x=6 to 7)</td>
<td>GPDMACxTR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DAO[12:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xA8+0x80 * x (x=6 to 7)</td>
<td>GPDMACxBF2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BRDAO[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xCB+0x80 * x (x=0 to 5)</td>
<td>GPDMACxLLR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>UTL[15:2]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xCC+0x80 * x (x=6 to 7)</td>
<td>GPDMACxLLR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>UTL[15:2]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reset values for each register are provided in the table.
Refer to *Section 2.3* for the register boundary addresses.
17 Nested vectored interrupt controller (NVIC)

17.1 NVIC main features

- 131 maskable interrupt channels (not including the 16 Cortex-M33 with FPU interrupt lines)
- 16 programmable priority levels (4 bits of interrupt priority used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

The NVIC and the processor core interface are closely coupled, enabling low-latency interrupt processing and efficient processing of late arriving interrupts.

The NVIC registers are banked across secure and non-secure states.
All interrupts including the core exceptions are managed by the NVIC.

17.2 SysTick calibration value register

The Cortex-M33 with TrustZone mainline security extension embeds two SysTick timers. When TrustZone is activated, the following SysTick timers are available:

- SysTick, secure instance
- SysTick, non-secure instance

When TrustZone is disabled, only one SysTick timer is available.

The SysTick timer calibration value (STCALIB) is 0x3E8. It gives a reference time base of 1 ms based on a SysTick clock frequency of 1 MHz. In order to match the 1 ms time base for an application running at a given frequency, the SysTick reload value must be programmed as follows in the SYST_RVR register:

- When SysTick clock source is CPU clock HCLK
  reload value = (HCLK x STCALIB) - 1

- When SysTick clock source is external clock (HCLK/8)
  reload value = ((HCLK/8) x STCALIB) - 1

The HCLK refers to the AHB frequency value in MHz.
Example: SysTick clock source is CPU clock HCLK of 100 MHz, to match a time base of 1 ms:
SysTick reload value = (100 x STCALIB) - 1 = 0x1869F

17.3 Interrupt and exception vectors

The grey rows in the following table describe the vectors without specific position.
Table 135. STM32H563/H573 and STM32H562 vector table

<table>
<thead>
<tr>
<th>Position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0000</td>
</tr>
<tr>
<td>-</td>
<td>-4</td>
<td>Fixed</td>
<td>SettableReset</td>
<td>Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.</td>
<td>0x0000 0004</td>
</tr>
<tr>
<td>-</td>
<td>-2</td>
<td>Fixed</td>
<td>NMI</td>
<td>Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.</td>
<td>0x0000 0008</td>
</tr>
<tr>
<td>-</td>
<td>-3 or -1</td>
<td>Fixed</td>
<td>Secure HardFault</td>
<td>Secure Hard fault</td>
<td>0x0000 000C</td>
</tr>
<tr>
<td>-</td>
<td>-1</td>
<td>Fixed</td>
<td>Non-secure HardFault</td>
<td>Non-Secure Hard fault. All classes of fault</td>
<td>0x0000 000C</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>Settable</td>
<td>MemManage</td>
<td>Memory management</td>
<td>0x0000 0010</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>Settable</td>
<td>BusFault</td>
<td>Pre-fetch fault, memory access fault</td>
<td>0x0000 0014</td>
</tr>
<tr>
<td>-</td>
<td>2</td>
<td>Settable</td>
<td>UsageFault</td>
<td>Undefined instruction or illegal state</td>
<td>0x0000 0018</td>
</tr>
<tr>
<td>-</td>
<td>3</td>
<td>Settable</td>
<td>SecureFault</td>
<td>Secure fault</td>
<td>0x0000 001C</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0020 - 0x0000 0028</td>
</tr>
<tr>
<td>-</td>
<td>4</td>
<td>-</td>
<td>SVC</td>
<td>System service call via SWI instruction</td>
<td>0x0000 002C</td>
</tr>
<tr>
<td>-</td>
<td>5</td>
<td>-</td>
<td>Debug Monitor</td>
<td>Monitor</td>
<td>0x0000 0030</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0034</td>
</tr>
<tr>
<td>-</td>
<td>6</td>
<td>Settable</td>
<td>PendSV</td>
<td>Pendable request for system service</td>
<td>0x0000 0038</td>
</tr>
<tr>
<td>-</td>
<td>7</td>
<td>Settable</td>
<td>SysTick</td>
<td>System tick timer</td>
<td>0x0000 003C</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>Settable</td>
<td>WWGD</td>
<td>Window watchdog interrupt</td>
<td>0x0000 0040</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>Settable</td>
<td>PVD_AVD</td>
<td>Power voltage monitor/ Analog voltage monitor</td>
<td>0x0000 0044</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>Settable</td>
<td>RTC</td>
<td>RTC global non-secure interrupts</td>
<td>0x0000 0048</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>Settable</td>
<td>RTC_S</td>
<td>RTC global secure interrupts</td>
<td>0x0000 004C</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>Settable</td>
<td>TAM</td>
<td>Tamper global interrupts</td>
<td>0x0000 0050</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>Settable</td>
<td>RAMCFG</td>
<td>RAM configuration global interrupt</td>
<td>0x0000 0054</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>Settable</td>
<td>FLASH</td>
<td>Flash non-secure global interrupt</td>
<td>0x0000 0058</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>Settable</td>
<td>FLASH_S</td>
<td>Flash secure global interrupt</td>
<td>0x0000 005C</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>Settable</td>
<td>GTZC</td>
<td>GTZC global interrupt</td>
<td>0x0000 0060</td>
</tr>
<tr>
<td>9</td>
<td>17</td>
<td>Settable</td>
<td>RCC</td>
<td>RCC non-secure global interrupt</td>
<td>0x0000 0064</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>Settable</td>
<td>RCC_S</td>
<td>RCC secure global interrupt</td>
<td>0x0000 0068</td>
</tr>
<tr>
<td>11</td>
<td>19</td>
<td>Settable</td>
<td>EXTI0</td>
<td>EXTI Line0 interrupt</td>
<td>0x0000 006C</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>Settable</td>
<td>EXTI1</td>
<td>EXTI Line1 interrupt</td>
<td>0x0000 0070</td>
</tr>
<tr>
<td>13</td>
<td>21</td>
<td>Settable</td>
<td>EXTI2</td>
<td>EXTI Line2 interrupt</td>
<td>0x0000 0074</td>
</tr>
</tbody>
</table>
Table 135. STM32H563/H573 and STM32H562 vector table (continued)

<table>
<thead>
<tr>
<th>Position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>22</td>
<td>Settable</td>
<td>EXTI3</td>
<td>EXTI Line3 interrupt</td>
<td>0x0000 0078</td>
</tr>
<tr>
<td>15</td>
<td>23</td>
<td>Settable</td>
<td>EXTI4</td>
<td>EXTI Line4 interrupt</td>
<td>0x0000 007C</td>
</tr>
<tr>
<td>16</td>
<td>24</td>
<td>Settable</td>
<td>EXTI5</td>
<td>EXTI Line5 interrupt</td>
<td>0x0000 0080</td>
</tr>
<tr>
<td>17</td>
<td>25</td>
<td>Settable</td>
<td>EXTI6</td>
<td>EXTI Line6 interrupt</td>
<td>0x0000 0084</td>
</tr>
<tr>
<td>18</td>
<td>26</td>
<td>Settable</td>
<td>EXTI7</td>
<td>EXTI Line7 interrupt</td>
<td>0x0000 0088</td>
</tr>
<tr>
<td>19</td>
<td>27</td>
<td>Settable</td>
<td>EXTI8</td>
<td>EXTI Line8 interrupt</td>
<td>0x0000 008C</td>
</tr>
<tr>
<td>20</td>
<td>28</td>
<td>Settable</td>
<td>EXTI9</td>
<td>EXTI Line9 interrupt</td>
<td>0x0000 0090</td>
</tr>
<tr>
<td>21</td>
<td>29</td>
<td>Settable</td>
<td>EXTI10</td>
<td>EXTI Line10 interrupt</td>
<td>0x0000 0094</td>
</tr>
<tr>
<td>22</td>
<td>30</td>
<td>Settable</td>
<td>EXTI11</td>
<td>EXTI Line11 interrupt</td>
<td>0x0000 0098</td>
</tr>
<tr>
<td>23</td>
<td>31</td>
<td>Settable</td>
<td>EXTI12</td>
<td>EXTI Line12 interrupt</td>
<td>0x0000 009C</td>
</tr>
<tr>
<td>24</td>
<td>32</td>
<td>Settable</td>
<td>EXTI13</td>
<td>EXTI Line13 interrupt</td>
<td>0x0000 00E4</td>
</tr>
<tr>
<td>25</td>
<td>33</td>
<td>Settable</td>
<td>EXTI14</td>
<td>EXTI Line14 interrupt</td>
<td>0x0000 00A0</td>
</tr>
<tr>
<td>26</td>
<td>34</td>
<td>Settable</td>
<td>EXTI15</td>
<td>EXTI Line15 interrupt</td>
<td>0x0000 00A4</td>
</tr>
<tr>
<td>27</td>
<td>35</td>
<td>Settable</td>
<td>GPDMA1_CH0</td>
<td>GPDMA1 channel 0 global interrupt</td>
<td>0x0000 00A8</td>
</tr>
<tr>
<td>28</td>
<td>36</td>
<td>Settable</td>
<td>GPDMA1_CH1</td>
<td>GPDMA1 channel 1 global interrupt</td>
<td>0x0000 00AC</td>
</tr>
<tr>
<td>29</td>
<td>37</td>
<td>Settable</td>
<td>GPDMA1_CH2</td>
<td>GPDMA1 channel 2 global interrupt</td>
<td>0x0000 00B0</td>
</tr>
<tr>
<td>30</td>
<td>38</td>
<td>Settable</td>
<td>GPDMA1_CH3</td>
<td>GPDMA1 channel 3 global interrupt</td>
<td>0x0000 00B4</td>
</tr>
<tr>
<td>31</td>
<td>39</td>
<td>Settable</td>
<td>GPDMA1_CH4</td>
<td>GPDMA1 channel 4 global interrupt</td>
<td>0x0000 00B8</td>
</tr>
<tr>
<td>32</td>
<td>40</td>
<td>Settable</td>
<td>GPDMA1_CH5</td>
<td>GPDMA1 channel 5 global interrupt</td>
<td>0x0000 00C0</td>
</tr>
<tr>
<td>33</td>
<td>41</td>
<td>Settable</td>
<td>GPDMA1_CH6</td>
<td>GPDMA1 channel 6 global interrupt</td>
<td>0x0000 00C4</td>
</tr>
<tr>
<td>34</td>
<td>42</td>
<td>Settable</td>
<td>GPDMA1_CH7</td>
<td>GPDMA1 channel 7 global interrupt</td>
<td>0x0000 00C8</td>
</tr>
<tr>
<td>35</td>
<td>43</td>
<td>Settable</td>
<td>IWDG</td>
<td>Independent watchdog interrupt</td>
<td>0x0000 00CC</td>
</tr>
<tr>
<td>36</td>
<td>44</td>
<td>Settable</td>
<td>SAES</td>
<td>Secure AES</td>
<td>0x0000 00D0</td>
</tr>
<tr>
<td>37</td>
<td>45</td>
<td>Settable</td>
<td>ADC1</td>
<td>ADC1 global interrupt</td>
<td>0x0000 00D4</td>
</tr>
<tr>
<td>38</td>
<td>46</td>
<td>Settable</td>
<td>DAC1</td>
<td>DAC1 global interrupt</td>
<td>0x0000 00D8</td>
</tr>
<tr>
<td>39</td>
<td>47</td>
<td>Settable</td>
<td>FDCAN1_IT0</td>
<td>FDCAN1 Interrupt 0</td>
<td>0x0000 00DC</td>
</tr>
<tr>
<td>40</td>
<td>48</td>
<td>Settable</td>
<td>FDCAN1_IT1</td>
<td>FDCAN1 Interrupt 1</td>
<td>0x0000 00E0</td>
</tr>
<tr>
<td>41</td>
<td>49</td>
<td>Settable</td>
<td>TIM1_BRK/ TIM1_TERR/ TIM1_IERR</td>
<td>TIM1 break/TIM1 transition error/TIM1 index error</td>
<td>0x0000 00E4</td>
</tr>
<tr>
<td>42</td>
<td>50</td>
<td>Settable</td>
<td>TIM1_UP</td>
<td>TIM1 update</td>
<td>0x0000 00E8</td>
</tr>
<tr>
<td>43</td>
<td>51</td>
<td>Settable</td>
<td>TIM1_TRG_COM/TIM1_DIR/TIM1_IDX</td>
<td>TIM1 trigger and commutation/TIM1 direction change interrupt/TIM1 index</td>
<td>0x0000 00EC</td>
</tr>
<tr>
<td>44</td>
<td>52</td>
<td>Settable</td>
<td>TIM1_CC</td>
<td>TIM1 capture compare interrupt</td>
<td>0x0000 00F0</td>
</tr>
</tbody>
</table>
Table 135. STM32H563/H573 and STM32H562 vector table (continued)

<table>
<thead>
<tr>
<th>Position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>53</td>
<td>Settable</td>
<td>TIM2</td>
<td>TIM2 global interrupt</td>
<td>0x0000 00F4</td>
</tr>
<tr>
<td>46</td>
<td>54</td>
<td>Settable</td>
<td>TIM3</td>
<td>TIM3 global interrupt</td>
<td>0x0000 00F8</td>
</tr>
<tr>
<td>47</td>
<td>55</td>
<td>Settable</td>
<td>TIM4</td>
<td>TIM4 global interrupt</td>
<td>0x0000 00FC</td>
</tr>
<tr>
<td>48</td>
<td>56</td>
<td>Settable</td>
<td>TIM5</td>
<td>TIM5 global interrupt</td>
<td>0x0000 0100</td>
</tr>
<tr>
<td>49</td>
<td>57</td>
<td>Settable</td>
<td>TIM6</td>
<td>TIM6 global interrupt</td>
<td>0x0000 0104</td>
</tr>
<tr>
<td>50</td>
<td>58</td>
<td>Settable</td>
<td>TIM7</td>
<td>TIM7 global interrupt</td>
<td>0x0000 0108</td>
</tr>
<tr>
<td>51</td>
<td>59</td>
<td>Settable</td>
<td>I2C1_EV</td>
<td>I2C1 event interrupt</td>
<td>0x0000 010C</td>
</tr>
<tr>
<td>52</td>
<td>60</td>
<td>Settable</td>
<td>I2C1_ER</td>
<td>I2C1 error interrupt</td>
<td>0x0000 0110</td>
</tr>
<tr>
<td>53</td>
<td>61</td>
<td>Settable</td>
<td>I2C2_EV</td>
<td>I2C2 event interrupt</td>
<td>0x0000 0114</td>
</tr>
<tr>
<td>54</td>
<td>62</td>
<td>Settable</td>
<td>I2C2_ER</td>
<td>I2C2 error interrupt</td>
<td>0x0000 0118</td>
</tr>
<tr>
<td>55</td>
<td>63</td>
<td>Settable</td>
<td>SPI1</td>
<td>SPI1 global interrupt</td>
<td>0x0000 011C</td>
</tr>
<tr>
<td>56</td>
<td>64</td>
<td>Settable</td>
<td>SPI2</td>
<td>SPI2 global interrupt</td>
<td>0x0000 0120</td>
</tr>
<tr>
<td>57</td>
<td>65</td>
<td>Settable</td>
<td>SPI3</td>
<td>SPI3 global interrupt</td>
<td>0x0000 0124</td>
</tr>
<tr>
<td>58</td>
<td>66</td>
<td>Settable</td>
<td>USART1</td>
<td>USART1 global interrupt</td>
<td>0x0000 0128</td>
</tr>
<tr>
<td>59</td>
<td>67</td>
<td>Settable</td>
<td>USART2</td>
<td>USART2 global interrupt</td>
<td>0x0000 012C</td>
</tr>
<tr>
<td>60</td>
<td>68</td>
<td>Settable</td>
<td>USART3</td>
<td>USART3 global interrupt</td>
<td>0x0000 0130</td>
</tr>
<tr>
<td>61</td>
<td>69</td>
<td>Settable</td>
<td>UART4</td>
<td>UART4 global interrupt</td>
<td>0x0000 0134</td>
</tr>
<tr>
<td>62</td>
<td>70</td>
<td>Settable</td>
<td>UART5</td>
<td>UART5 global interrupt</td>
<td>0x0000 0138</td>
</tr>
<tr>
<td>63</td>
<td>71</td>
<td>Settable</td>
<td>LPUART1</td>
<td>LPUART1 global interrupt or LPUART1 R wakeup or LPUART1 T wakeup through EXTI line</td>
<td>0x0000 013C</td>
</tr>
<tr>
<td>64</td>
<td>72</td>
<td>Settable</td>
<td>LPTIM1 OR LPTIM1_AIT</td>
<td>LPTIM1 global interrupt or LPTimer1 AIT through EXTI line</td>
<td>0x0000 0140</td>
</tr>
<tr>
<td>65</td>
<td>73</td>
<td>Settable</td>
<td>TIM8_BRK/TIM8_TERR/TIM8_IERR</td>
<td>TIM8 break interrupt/TIM8 transition error/TIM8 index error</td>
<td>0x0000 0144</td>
</tr>
<tr>
<td>66</td>
<td>74</td>
<td>Settable</td>
<td>TIM8_UP</td>
<td>TIM8 update interrupt</td>
<td>0x0000 0148</td>
</tr>
<tr>
<td>67</td>
<td>75</td>
<td>Settable</td>
<td>TIM8_TRG_COM/TIM8_DIR/TIM8_IDX</td>
<td>TIM8 trigger and commutation interrupt/TIM8 direction change interrupt/TIM8 index</td>
<td>0x0000 014C</td>
</tr>
<tr>
<td>68</td>
<td>76</td>
<td>Settable</td>
<td>TIM8_CC</td>
<td>TIM8 capture compare interrupt</td>
<td>0x0000 0150</td>
</tr>
<tr>
<td>69</td>
<td>77</td>
<td>Settable</td>
<td>ADC2</td>
<td>ADC2 global interrupt</td>
<td>0x0000 0154</td>
</tr>
<tr>
<td>70</td>
<td>78</td>
<td>Settable</td>
<td>LPTIM2 OR LPTIM2_AIT</td>
<td>LPTIM2 global interrupt or LPTimer2 AIT through EXTI line</td>
<td>0x0000 0158</td>
</tr>
<tr>
<td>71</td>
<td>79</td>
<td>Settable</td>
<td>TIM15</td>
<td>TIM15 global interrupt</td>
<td>0x0000 015C</td>
</tr>
</tbody>
</table>
### Table 135. STM32H563/H573 and STM32H562 vector table (continued)

<table>
<thead>
<tr>
<th>Position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>80</td>
<td>Settable</td>
<td>TIM16</td>
<td>TIM16 global interrupt</td>
<td>0x0000 0160</td>
</tr>
<tr>
<td>73</td>
<td>81</td>
<td>Settable</td>
<td>TIM17</td>
<td>TIM17 global interrupt</td>
<td>0x0000 0164</td>
</tr>
<tr>
<td>74</td>
<td>82</td>
<td>Settable</td>
<td>USB_FS</td>
<td>USB FS global interrupt</td>
<td>0x0000 0168</td>
</tr>
<tr>
<td>75</td>
<td>83</td>
<td>Settable</td>
<td>CRS</td>
<td>Clock recovery system global interrupt</td>
<td>0x0000 016C</td>
</tr>
<tr>
<td>76</td>
<td>84</td>
<td>Settable</td>
<td>UCPD1</td>
<td>UCPD1 global interrupt</td>
<td>0x0000 0170</td>
</tr>
<tr>
<td>77</td>
<td>85</td>
<td>Settable</td>
<td>FMC</td>
<td>FMC global interrupt</td>
<td>0x0000 0174</td>
</tr>
<tr>
<td>78</td>
<td>86</td>
<td>Settable</td>
<td>OCTOSPI1</td>
<td>OCTOSPI1 global interrupt</td>
<td>0x0000 0178</td>
</tr>
<tr>
<td>79</td>
<td>87</td>
<td>Settable</td>
<td>SDMMC1</td>
<td>SDMMC1 global interrupt</td>
<td>0x0000 017C</td>
</tr>
<tr>
<td>80</td>
<td>88</td>
<td>Settable</td>
<td>I2C3_EV</td>
<td>I2C3 event interrupt</td>
<td>0x0000 0180</td>
</tr>
<tr>
<td>81</td>
<td>89</td>
<td>Settable</td>
<td>I2C3_ER</td>
<td>I2C3 error interrupt</td>
<td>0x0000 0184</td>
</tr>
<tr>
<td>82</td>
<td>90</td>
<td>Settable</td>
<td>SPI4</td>
<td>SPI4 global interrupt</td>
<td>0x0000 0188</td>
</tr>
<tr>
<td>83</td>
<td>91</td>
<td>Settable</td>
<td>SPI5</td>
<td>SPI5 global interrupt</td>
<td>0x0000 0174</td>
</tr>
<tr>
<td>84</td>
<td>92</td>
<td>Settable</td>
<td>SPI6</td>
<td>SPI6 global interrupt</td>
<td>0x0000 0178</td>
</tr>
<tr>
<td>85</td>
<td>93</td>
<td>Settable</td>
<td>USART6</td>
<td>USART6 global interrupt</td>
<td>0x0000 017C</td>
</tr>
<tr>
<td>86</td>
<td>94</td>
<td>Settable</td>
<td>USART10</td>
<td>USART10 global interrupt</td>
<td>0x0000 0180</td>
</tr>
<tr>
<td>87</td>
<td>95</td>
<td>Settable</td>
<td>USART11</td>
<td>USART11 global interrupt</td>
<td>0x0000 0184</td>
</tr>
<tr>
<td>88</td>
<td>96</td>
<td>Settable</td>
<td>SAI1</td>
<td>SAI1 global interrupt</td>
<td>0x0000 0188</td>
</tr>
<tr>
<td>89</td>
<td>97</td>
<td>Settable</td>
<td>SAI2</td>
<td>SAI2 global interrupt</td>
<td>0x0000 018C</td>
</tr>
<tr>
<td>90</td>
<td>98</td>
<td>Settable</td>
<td>GPDMA2_CH0</td>
<td>GPDMA2 channel0 global interrupt</td>
<td>0x0000 0190</td>
</tr>
<tr>
<td>91</td>
<td>99</td>
<td>Settable</td>
<td>GPDMA2_CH1</td>
<td>GPDMA2 channel1 global interrupt</td>
<td>0x0000 0194</td>
</tr>
<tr>
<td>92</td>
<td>100</td>
<td>Settable</td>
<td>GPDMA2_CH2</td>
<td>GPDMA2 channel2 global interrupt</td>
<td>0x0000 0198</td>
</tr>
<tr>
<td>93</td>
<td>101</td>
<td>Settable</td>
<td>GPDMA2_CH3</td>
<td>GPDMA2 channel3 global interrupt</td>
<td>0x0000 019C</td>
</tr>
<tr>
<td>94</td>
<td>102</td>
<td>Settable</td>
<td>GPDMA2_CH4</td>
<td>GPDMA2 channel4 global interrupt</td>
<td>0x0000 01A0</td>
</tr>
<tr>
<td>95</td>
<td>103</td>
<td>Settable</td>
<td>GPDMA2_CH5</td>
<td>GPDMA2 channel5 global interrupt</td>
<td>0x0000 01A4</td>
</tr>
<tr>
<td>96</td>
<td>104</td>
<td>Settable</td>
<td>GPDMA2_CH6</td>
<td>GPDMA2 channel6 global interrupt</td>
<td>0x0000 01A8</td>
</tr>
<tr>
<td>97</td>
<td>105</td>
<td>Settable</td>
<td>GPDMA2_CH7</td>
<td>GPDMA2 channel7 global interrupt</td>
<td>0x0000 01AC</td>
</tr>
<tr>
<td>98</td>
<td>106</td>
<td>Settable</td>
<td>UART7</td>
<td>UART7 global interrupt</td>
<td>0x0000 01B0</td>
</tr>
<tr>
<td>99</td>
<td>107</td>
<td>Settable</td>
<td>UART8</td>
<td>UART8 global interrupt</td>
<td>0x0000 01B4</td>
</tr>
<tr>
<td>100</td>
<td>108</td>
<td>Settable</td>
<td>UART9</td>
<td>UART9 global interrupt</td>
<td>0x0000 01B8</td>
</tr>
<tr>
<td>101</td>
<td>109</td>
<td>Settable</td>
<td>UART12</td>
<td>UART12 global interrupt</td>
<td>0x0000 01BC</td>
</tr>
<tr>
<td>102</td>
<td>110</td>
<td>Settable</td>
<td>SDMMC2</td>
<td>SDMMC2 global interrupt</td>
<td>0x0000 01C0</td>
</tr>
<tr>
<td>103</td>
<td>111</td>
<td>Settable</td>
<td>FPU</td>
<td>Floating point interrupt</td>
<td>0x0000 01C4</td>
</tr>
<tr>
<td>104</td>
<td>112</td>
<td>Settable</td>
<td>ICACHE</td>
<td>Instruction cache global interrupt</td>
<td>0x0000 01C8</td>
</tr>
</tbody>
</table>
### Table 135. STM32H563/H573 and STM32H562 vector table (continued)

<table>
<thead>
<tr>
<th>Position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>105</td>
<td>113</td>
<td>Settable</td>
<td>DCACHE</td>
<td>Data cache global interrupt</td>
<td>0x0000 01CC</td>
</tr>
<tr>
<td>106</td>
<td>114</td>
<td>Settable</td>
<td>ETH</td>
<td>Ethernet interrupt</td>
<td>0x0000 01D0</td>
</tr>
<tr>
<td>107</td>
<td>115</td>
<td>Settable</td>
<td>ETH_WKUP</td>
<td>ETHERNET wakeup interrupt through EXTI line</td>
<td>0x0000 01D4</td>
</tr>
<tr>
<td>108</td>
<td>116</td>
<td>Settable</td>
<td>DCMI_PSSI</td>
<td>DCMI/PSSI global interrupt</td>
<td>0x0000 01D8</td>
</tr>
<tr>
<td>109</td>
<td>117</td>
<td>Settable</td>
<td>FDCAN2_IT0</td>
<td>FDCAN2 interrupt 0</td>
<td>0x0000 01DC</td>
</tr>
<tr>
<td>110</td>
<td>118</td>
<td>Settable</td>
<td>FDCAN2_IT1</td>
<td>FDCAN2 interrupt 1</td>
<td>0x0000 01E0</td>
</tr>
<tr>
<td>111</td>
<td>119</td>
<td>Settable</td>
<td>CORDIC</td>
<td>CORDIC interrupt</td>
<td>0x0000 01E4</td>
</tr>
<tr>
<td>112</td>
<td>120</td>
<td>Settable</td>
<td>FMAC</td>
<td>FMAC interrupt</td>
<td>0x0000 01E8</td>
</tr>
<tr>
<td>113</td>
<td>121</td>
<td>Settable</td>
<td>DTS OR DTS_WKUP</td>
<td>DTS interrupt or DTS AIT through EXTI line</td>
<td>0x0000 01EC</td>
</tr>
<tr>
<td>114</td>
<td>122</td>
<td>Settable</td>
<td>RNG</td>
<td>RNG global interrupt</td>
<td>0x0000 01F0</td>
</tr>
<tr>
<td>115</td>
<td>123</td>
<td>Settable</td>
<td>OTFDEC1</td>
<td>OTFDEC1 secure global interrupt</td>
<td>0x0000 01F4</td>
</tr>
<tr>
<td>116</td>
<td>124</td>
<td>Settable</td>
<td>AES</td>
<td>AES global interrupt</td>
<td>0x0000 01F8</td>
</tr>
<tr>
<td>117</td>
<td>125</td>
<td>Settable</td>
<td>HASH</td>
<td>HASH interrupt</td>
<td>0x0000 01FC</td>
</tr>
<tr>
<td>118</td>
<td>126</td>
<td>Settable</td>
<td>PKA</td>
<td>PKA global interrupt</td>
<td>0x0000 0200</td>
</tr>
<tr>
<td>119</td>
<td>127</td>
<td>Settable</td>
<td>CEC</td>
<td>HDMI-CEC global interrupt</td>
<td>0x0000 0204</td>
</tr>
<tr>
<td>120</td>
<td>128</td>
<td>Settable</td>
<td>TIM12</td>
<td>TIM12 global interrupt</td>
<td>0x0000 0208</td>
</tr>
<tr>
<td>121</td>
<td>129</td>
<td>Settable</td>
<td>TIM13</td>
<td>TIM13 global interrupt</td>
<td>0x0000 020C</td>
</tr>
<tr>
<td>122</td>
<td>130</td>
<td>Settable</td>
<td>TIM14</td>
<td>TIM14 global interrupt</td>
<td>0x0000 0210</td>
</tr>
<tr>
<td>123</td>
<td>131</td>
<td>Settable</td>
<td>I3C1_EV</td>
<td>I3C1 event interrupt</td>
<td>0x0000 0214</td>
</tr>
<tr>
<td>124</td>
<td>132</td>
<td>Settable</td>
<td>I3C1_ER</td>
<td>I3C1 error interrupt</td>
<td>0x0000 0218</td>
</tr>
<tr>
<td>125</td>
<td>133</td>
<td>Settable</td>
<td>I2C4_EV</td>
<td>I2C4 event interrupt</td>
<td>0x0000 021C</td>
</tr>
<tr>
<td>126</td>
<td>134</td>
<td>Settable</td>
<td>I2C4_ER</td>
<td>I2C4 error interrupt</td>
<td>0x0000 0220</td>
</tr>
<tr>
<td>127</td>
<td>135</td>
<td>Settable</td>
<td>LPTIM3 OR LPTIM3_AIT</td>
<td>LPTIM3 global interrupt or LPTimer3 AIT through EXTI line</td>
<td>0x0000 0224</td>
</tr>
<tr>
<td>128</td>
<td>136</td>
<td>Settable</td>
<td>LPTIM4 OR LPTIM4_AIT</td>
<td>LPTIM4 global interrupt or LPTimer4 AIT through EXTI line</td>
<td>0x0000 0228</td>
</tr>
<tr>
<td>129</td>
<td>137</td>
<td>Settable</td>
<td>LPTIM5 OR LPTIM5_AIT</td>
<td>LPTIM5 global interrupt or LPTimer5 AIT through EXTI line</td>
<td>0x0000 022C</td>
</tr>
<tr>
<td>130</td>
<td>138</td>
<td>Settable</td>
<td>LPTIM6 OR LPTIM6_AIT</td>
<td>LPTIM6 global interrupt or LPTimer6 AIT through EXTI line</td>
<td>0x0000 0230</td>
</tr>
</tbody>
</table>
18 Extended interrupts and event controller (EXTI)

The extended interrupts and event controller (EXTI) manages the individual CPU and system wakeup through configurable and direct event inputs. It provides wakeup requests to the power control and generates an interrupt request to the CPU NVIC and events to the CPU event input. For the CPU, an additional event generation block (EVG) is needed to generate the CPU event signal.

The EXTI wakeup requests allow the system to be woken up from Stop modes.

The interrupt request and event request generation can be used also in Run modes.

The EXTI also includes the EXTI mux IO port selection.

18.1 EXTI main features

The EXTI main features are the following:

- 58 input events supported
- All event inputs allow the possibility to wake up the system.
- Events that do not have an associated wakeup flag in the peripheral, have a flag in the EXTI and generate an interrupt to the CPU from the EXTI.
- Events can be used to generate a CPU wakeup event.

The asynchronous event inputs are classified in two groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse), with the following features
  - Selectable active trigger edge
  - Interrupt pending status register bits independent for the rising and falling edge
  - Individual interrupt and event generation mask, used for conditioning the CPU wakeup, interrupt and event generation
  - Software trigger possibility
  - Secure events: The access to control and configuration bits of secure input events can be made secure and or privilege.
  - EXTI IO port selection

- Direct events (interrupt and wake-up sources from peripherals having an associated flag which requires to be cleared in the peripheral), with the following features:
  - Fixed rising edge active trigger
  - No interrupt pending status register bit in the EXTI (the interrupt pending status flag is provided by the peripheral generating the event)
  - Individual interrupt and event generation mask, used to condition the CPU wake-up and event generation
  - No software trigger possibility

18.2 EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input trigger block, the masking block and EXTI mux as shown in Figure 87.
The register block contains all the EXTI registers.

The event input trigger block provides event input edge trigger logic.

The masking block provides the event input distribution to the different wakeup, interrupt and event outputs, and their masking.

The EXTI mux provides the IO port selection on to the EXTI event signal.

**Figure 87. EXTI block diagram**

<table>
<thead>
<tr>
<th>Pin name</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHB interface</td>
<td>I/O</td>
<td>EXTI register bus interface. When one event is configured to enable security, the AHB interface supports secure accesses.</td>
</tr>
<tr>
<td>hclk</td>
<td>I</td>
<td>AHB bus clock and EXTI system clock</td>
</tr>
<tr>
<td>Configurable event(y)</td>
<td>I</td>
<td>Asynchronous wake-up events from peripherals without an associated interrupt and flag</td>
</tr>
<tr>
<td>Direct event(x)</td>
<td>I</td>
<td>Synchronous and asynchronous wake-up events from peripherals with an associated interrupt and flag</td>
</tr>
<tr>
<td>exti_ilac</td>
<td>O</td>
<td>Illegal access event</td>
</tr>
<tr>
<td>IOPort(n)</td>
<td>I</td>
<td>GPIOs block IO ports[15:0]</td>
</tr>
<tr>
<td>exti[15:0]</td>
<td>O</td>
<td>EXTI GPIO output port to trigger other peripherals</td>
</tr>
<tr>
<td>it_exti_per(y)</td>
<td>O</td>
<td>Interrupts to the CPU associated with configurable event (y)</td>
</tr>
<tr>
<td>c_evt_exti</td>
<td>O</td>
<td>High-level sensitive event output for CPU, synchronous to hclk</td>
</tr>
<tr>
<td>c_evt_rst</td>
<td>I</td>
<td>Asynchronous reset input to clear c_evt_exti</td>
</tr>
<tr>
<td>sys_wakeup</td>
<td>O</td>
<td>Asynchronous system wakeup request to PWR for ck_sys and hclk</td>
</tr>
<tr>
<td>c_wakeup</td>
<td>O</td>
<td>Wakeup request to PWR for CPU, synchronous to hclk</td>
</tr>
</tbody>
</table>
### EXTI connections between peripherals and CPU

Some peripherals able to generate wake-up or interrupt events when the system is in Stop mode, are connected to the EXTI.

- Peripheral wakeup signals that generate a pulse or do not have an interrupt status bits in the peripheral, are connected to an EXTI configurable event input. For these events, the EXTI provides a status pending bit to be cleared. It is the EXTI interrupt, associated with the status bit, that interrupts the CPU.
- Peripheral interrupt and wake-up signals with a status bit in the peripheral to be cleared are connected to an EXTI direct event input. There is no status pending bit within the EXTI. The interrupt or wake-up is cleared by the CPU in the peripheral. It is the peripheral interrupt that interrupts the CPU directly.

All GPIO ports input to the EXTI multiplexer allow the selection of a port pin to wake up the system via a configurable event.

The EXTI configurable event interrupts are connected to the NVIC.

The dedicated EXTI/EVG CPU event is connected to the CPU rxev input.

The EXTI CPU wakeup signals are connected to the PWR and are used to wake up the system and the CPU sub-system bus clocks.

### EXTI interrupt/event mapping

The EXTI lines are connected as shown in the table below.

<table>
<thead>
<tr>
<th>EXTI Line</th>
<th>Line source</th>
<th>Line type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-15</td>
<td>GPIO</td>
<td>Configurable</td>
</tr>
<tr>
<td>16</td>
<td>PVD/AVD output</td>
<td>Configurable</td>
</tr>
<tr>
<td>17</td>
<td>RTC non-secure</td>
<td>-</td>
</tr>
<tr>
<td>18</td>
<td>RTC secure</td>
<td>-</td>
</tr>
<tr>
<td>19</td>
<td>TAMP non-secure</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>TAMP secure</td>
<td>-</td>
</tr>
<tr>
<td>21</td>
<td>I2C1 wakeup</td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>I2C2 wakeup</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>I2C3 wakeup</td>
<td>-</td>
</tr>
<tr>
<td>24</td>
<td>I3C wakeup</td>
<td>-</td>
</tr>
<tr>
<td>EXTI Line</td>
<td>Line source</td>
<td>Line type</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>25</td>
<td>USART1 wakeup</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>USART2 wakeup</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>USART3 wakeup</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>UART4 wakeup</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>UART5 wakeup</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>USART6 wakeup</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>UART7 wakeup</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>UART8 wakeup</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>UART9 wakeup</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>USART10 wakeup</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>USART11 wakeup</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>UART12 wakeup</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>LPUART1 wakeup</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>LPTIM1</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>LPTIM2</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>SPI1 wakeup</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>SPI2 wakeup</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>SPI3 wakeup</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>SPI4 wakeup</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>SPI5 wakeup</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>SPI6 wakeup</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>ETH wakeup</td>
<td>Configurable</td>
</tr>
<tr>
<td>47</td>
<td>USB FS wakeup</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>USBPD1 wakeup</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>LPTIM2 CH1</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>DTS wakeup</td>
<td>Configurable</td>
</tr>
<tr>
<td>51</td>
<td>HDMI-CEC wakeup</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>I2C4 wakeup</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>UVM output</td>
<td>Configurable</td>
</tr>
<tr>
<td>54</td>
<td>LPTIM3</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>LPTIM4</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>LPTIM5</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>LPTIM6</td>
<td></td>
</tr>
</tbody>
</table>
18.3 EXTI functional description

The events features are controlled from register bits as follows:

- **Active trigger edge enable**
  - by rising edge selection in the *EXTI rising trigger selection register* (EXTI_RTSR1) and *EXTI rising trigger selection register 2* (EXTI_RTSR2)
  - by falling edge selection in the *EXTI falling trigger selection register* (EXTI_FTSR1) and *EXTI falling trigger selection register 2* (EXTI_FTSR2)

- **Software trigger** in the *EXTI software interrupt event register* (EXTI_SWIER1) and *EXTI software interrupt event register 2* (EXTI_SWIER2)

- **Interrupt pending flag** in the
  - *EXTI rising edge pending register* (EXTI_RPR1) and *EXTI rising edge pending register 2* (EXTI_RPR2)
  - *EXTI falling edge pending register* (EXTI_FPR1) and *EXTI falling edge pending register 2* (EXTI_FPR2)

- **CPU wakeup and interrupt enable** in the
  - *EXTI CPU wakeup with interrupt mask register* (EXTI_IMR1) and *EXTI CPU wakeup with interrupt mask register 2* (EXTI_IMR2)

- **CPU wakeup and event enable**
  - *EXTI CPU wakeup with event mask register* (EXTI_EMR1) and *EXTI CPU wakeup with event mask register 2* (EXTI_EMR2)

18.3.1 EXTI configurable event input wakeup

The figure below is a detailed representation of the logic associated with configurable event inputs that wake up the CPU sub-system bus clocks and generate an EXTI pending flag and interrupt to the CPU, and/or a CPU wakeup event.

Figure 88. Configurable event trigger logic CPU wakeup

1. Only for the input events that support CPU rxev generation c_event.

The software interrupt event register allows configurable events to be triggered by software, writing the corresponding register bit, whatever the edge selection setting.
The configurable event active trigger edge (or both edges) is selected and enabled in the rising/falling edge selection registers.

The CPU has its dedicated wakeup (interrupt) mask register and a dedicated event mask registers. When the event is enabled, it is generated to the CPU. All events for the CPU are ORed together into a single CPU event signal. The event pending registers (EXTI_RPR and EXTI_FPR) are not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to the CPU. The configurable event interrupts must be acknowledged by software in the EXTI_RPR and/or EXTI_FPR registers.

When a CPU wakeup (interrupt) or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

**Note:** A detected configurable event interrupt pending request can be cleared by the CPU with the correct access permission. The system is not able to enter into low-power modes as long as an interrupt pending request is active.

### 18.3.2 EXTI direct event input wake-up

The direct events do not have an associated EXTI interrupt. The EXTI only wakes up the system and CPU subsystem clocks, and can generate a CPU wake-up event. The peripheral synchronous interrupt, associated with the direct wake-up event, wakes up the CPU.

The EXTI direct event is able to generate a CPU event that wakes up the CPU. The CPU event may occur before the associated peripheral interrupt flag is set.

**Figure 89. EXTI direct events**

### 18.3.3 EXTI mux selection

The EXTI mux allows the selection of GPIOs as interrupts and wakeup. GPIOs are connected via 16 EXTI mux lines to the first 16 EXTI events as configurable event. The
selection of GPIO port as EXTI mux output is controlled in the EXTI external interrupt selection register (EXTI_EXTICR1).

Figure 90. EXTI mux GPIO selection

The EXTI mux outputs are available as output signals from the EXTI to trigger other peripherals, whatever the masking in EXTI_IMR and EXTI_EMR registers.

18.4 EXTI functional behavior

The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the CPU wakeup generation is conditioned by the CPU interrupt mask and CPU event mask.

Table 139. Masking functionality

<table>
<thead>
<tr>
<th>CPU interrupt enable (in EXTI_IMR.IMn)</th>
<th>CPU event enable (in EXTI_EMR.EMn)</th>
<th>Configurable event inputs (in EXTI_RPR.RPIFn and EXTI_FPR.FPIFn)</th>
<th>Exti(n) interrupt(1)</th>
<th>CPU event</th>
<th>CPU wakeup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>No</td>
<td>Masked</td>
<td>Masked</td>
<td>Masked</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>No</td>
<td>Masked</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Status latched</td>
<td>Yes</td>
<td>Masked</td>
<td>Yes(2)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Status latched</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

1. The single exti(n) interrupt goes to the CPU. If no interrupt is required for CPU(m), the exti(n) interrupt must be masked in the CPU NVIC.
2. Only if CPU interrupt is enabled in EXTI_IMR.IMn.

For configurable event inputs, when the enabled edges occur on the event input, an event request is generated. When the associated CPU interrupt is unmasked, the corresponding pending bits EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn is/are set: the CPU sub-system is woken up and the CPU interrupt signal is activated. The EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn pending bits must be cleared by software writing it to 1. This action clears the CPU interrupt.

For the configurable event inputs, an event request can be generated by software when writing a 1 in the software interrupt/event register EXTI_SWIER, allowing the generation of a
rising edge on the event. The rising edge event pending bit is set in EXTI_RPR, whatever the setting in EXTI_RTSR.

18.5 EXTI event protection

The EXTI is able to protect event register bits from being modified by non-secure and unprivileged accesses. The protection is individually activated per input event via the register bits in EXTI_SECCFG and EXTI_PRIVCFG. At EXTI level, the protection consists in preventing the following unauthorized write access:

- Change the settings of the secure and/or privileged configurable events.
- Change the masking of the secure and/or privileged input events.
- Clear pending status of the secure and/or privileged input events.

<table>
<thead>
<tr>
<th>Register name</th>
<th>Access type</th>
<th>Protection(1)(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTI_RTSR</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>EXTI_FTSR</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>EXTI_SWIER</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>EXTI_RPR</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>EXTI_FPR</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>EXTI_SECCFG</td>
<td>RW</td>
<td>Security and privilege can be bit-wise enabled in EXTI_SECCFG and EXTI_PRIVCFG.</td>
</tr>
<tr>
<td>EXTI_PRIVCFG</td>
<td>RW</td>
<td>Security and privilege can be bit-wise enabled in EXTI_SECCFG and EXTI_PRIVCFG.</td>
</tr>
<tr>
<td>EXTI_EXTICRn</td>
<td>RW</td>
<td>Security and privilege can be bit-wise enabled in EXTI_SECCFG and EXTI_PRIVCFG.</td>
</tr>
<tr>
<td>EXTI_LOCKR</td>
<td>RW</td>
<td>Always secure</td>
</tr>
<tr>
<td>EXTI_IM</td>
<td>RW</td>
<td>Security and privilege can be bit-wise enabled in EXTI_SECCFG and EXTI_PRIVCFG.</td>
</tr>
<tr>
<td>EXTI_EM</td>
<td>RW</td>
<td></td>
</tr>
</tbody>
</table>

1. Security is enabled with the individual input event (EXTI_SECCFG register).
2. Privilege is enabled with the individual Input event (EXTI_PRIVCFG register).

18.5.1 EXTI security protection

When security is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a secure access. A non-secure write access is discarded and a read returns 0.

When input events are non-secure, the security is disabled. The associated input event configuration and control bits can be modified and read by a secure access and non-secure access.

The security configuration in registers EXTI_SECCFG can be globally locked after reset by EXTI_LOCKR.LOCK.
18.5.2 EXTI privilege protection

When privilege is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a privileged access. An unprivileged write access is discarded and a read returns 0.

When input events are unprivileged, the privilege is disabled. The associated input event configuration and control bits can be modified and read by a privileged access and unprivileged access.

The privileged configuration in registers EXTI_PRIVCFGR can be globally locked after reset by EXTI_LOCKR.LOCK.
18.6 EXTI registers

The EXTI register map is divided in the following sections:

Table 141. EXTI register map sections

<table>
<thead>
<tr>
<th>Address offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000 - 0x01C</td>
<td>General configurable event [31:0] configuration</td>
</tr>
<tr>
<td>0x020 - 0x03C</td>
<td>General configurable event [57:32] configuration</td>
</tr>
<tr>
<td>0x060 - 0x06C</td>
<td>EXTI IO port mux selection</td>
</tr>
<tr>
<td>0x070</td>
<td>EXTI protection lock configuration</td>
</tr>
<tr>
<td>0x080 - 0x0BC</td>
<td>CPU input event configuration</td>
</tr>
</tbody>
</table>

All the registers can be accessed with word (32-bit), half-word (16-bit) and byte (8-bit) access.

18.6.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x000
Reset value: 0x0000 0000
Contains only register bits for configurable events.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RT16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RT15</td>
<td>RT14</td>
<td>RT13</td>
<td>RT12</td>
<td>RT11</td>
<td>RT10</td>
<td>RT9</td>
<td>RT8</td>
<td>RT7</td>
<td>RT6</td>
<td>RT5</td>
<td>RT4</td>
<td>RT3</td>
<td>RT2</td>
<td>RT1</td>
<td>RT0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bits 16:0 **RTx**: Rising trigger event configuration bit of configurable event input x\(^{(1)}\) (x = 16 to 0)
- When EXTI
  - SECCFRG.RSECx is disabled, RTx can be accessed with non-secure and secure access.
  - SECCFRG.RSECx is enabled, RTx can only be accessed with secure access. Non-secure write to this bit x is discarded and non-secure read returns 0.
  - PRIVCFRG.PRIVx is disabled, RTx can be accessed with unprivileged and privileged access.
  - PRIVCFRG.PRIVx is enabled, RTx can only be accessed with privileged access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.
- 0: Rising trigger disabled (for event and interrupt) for input line
- 1: Rising trigger enabled (for event and interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

18.6.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x004
Reset value: 0x0000 0000
Contains only register bits for configurable events.

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

<table>
<thead>
<tr>
<th>Bits 16:0</th>
<th>FTx: Falling trigger event configuration bit of configurable event input x (x = 16 to 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with non-secure and secure access.</td>
</tr>
<tr>
<td></td>
<td>When EXTI_SECCFGR.SECx is enabled, FTx can only be accessed with secure access. Non-secure write to this FTx is discarded, non-secure read returns 0. When EXTI_PRIVCFGPR.PRIVx is disabled, FTx can be accessed with unprivileged and privileged access. When EXTI_PRIVCFGPR.PRIVx is enabled, FTx can only be accessed with privileged access. Unprivileged write to this FTx is discarded, unprivileged read returns 0. 0: Falling trigger disabled (for event and Interrupt) for input line 1: Falling trigger enabled (for event and Interrupt) for input line.</td>
</tr>
</tbody>
</table>

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

18.6.3 EXTI software interrupt event register (EXTI_SWIER1)
Address offset: 0x008
Reset value: 0x0000 0000
Contains only register bits for configurable events.

| Bits 31:17 | Reserved, must be kept at reset value. |
Bits 16:0 **SWIx**: Software interrupt on event x (x = 16 to 0)

When EXTI_SECCFGGR.SECx is disabled, SWIx can be accessed with non-secure and secure access.

When EXTI_SECCFGGR.SECx is enabled, SWIx can only be accessed with secure access.

Non-secure write to this SWIx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGGR.PRIVx is enabled, SWIx can only be accessed with privileged access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

0: Writing 0 has no effect.
1: Writing 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.

### 18.6.4 EXTI rising edge pending register (EXTI_RPR1)

Address offset: 0x00C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPIF0</td>
<td>RPIF1</td>
<td>RPIF2</td>
<td>RPIF3</td>
<td>RPIF4</td>
<td>RPIF5</td>
<td>RPIF6</td>
<td>RPIF7</td>
<td>RPIF8</td>
<td>RPIF9</td>
<td>RPIF10</td>
<td>RPIF11</td>
<td>RPIF12</td>
<td>RPIF13</td>
<td>RPIF14</td>
<td>RPIF15</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bits 16:0 **RPIFx**: configurable event inputs x rising edge pending bit (x = 16 to 0)

When EXTI_SECCFGGR.SECx is disabled, RPIFx can be accessed with non-secure and secure access.

When EXTI_SECCFGGR.SECx is enabled, RPIFx can only be accessed with secure access.

Non-secure write to this RPIFx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGGR.PRIVx is enabled, RPIFx can only be accessed with privileged access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.

0: No rising edge trigger request occurred
1: Rising edge trigger request occurred

This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.
### 18.6.5 EXTI falling edge pending register (EXTI_FPR1)

Address offset: 0x010  
Reset value: 0x0000 0000  
Contains only register bits for configurable events.

<table>
<thead>
<tr>
<th>Bit 31:17 Reserved, must be kept at reset value.</th>
<th>Bit 16:0 FPIFx: configurable event inputs x falling edge pending bit (x = 16 to 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:17 Reserved, must be kept at reset value.</td>
<td>When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with non-secure and secure access.</td>
</tr>
<tr>
<td>Bits 16:0 FPIFx: configurable event inputs x falling edge pending bit (x = 16 to 0)</td>
<td>When EXTI_SECCFGR.SECx is enabled, FPIFx can only be accessed with secure access. Non-secure write to this FPIFx is discarded, non-secure read returns 0.</td>
</tr>
<tr>
<td>When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privileged access.</td>
<td>When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can only be accessed with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.</td>
</tr>
<tr>
<td>0: No falling edge trigger request occurred</td>
<td>1: Falling edge trigger request occurred</td>
</tr>
<tr>
<td>This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.</td>
<td></td>
</tr>
</tbody>
</table>

### 18.6.6 EXTI security configuration register (EXTI_SECCFGR1)

Address offset: 0x014  
Reset value: 0x0000 0000  
This register provides write access security, a non-secure write access is ignored and causes the generation of an illegal access event. A non-secure read returns the register data.

Contains only register bits for security capable input events.

<table>
<thead>
<tr>
<th>Bit 31:17 Reserved, must be kept at reset value.</th>
<th>Bit 15:0 SECx: security capable input x (x = 16 to 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:17 Reserved, must be kept at reset value.</td>
<td>When EXTI_PRIVCFGR.PRIVx is disabled, SECx can be accessed with unprivileged and privileged access.</td>
</tr>
<tr>
<td>Bits 15:0 SECx: security capable input x (x = 16 to 0)</td>
<td>When EXTI_PRIVCFGR.PRIVx is enabled, SECx can only be accessed with privileged access. Unprivileged write to this SECx is discarded, unprivileged read returns 0.</td>
</tr>
<tr>
<td>0: No illegal access triggers due to a non-privileged write or read</td>
<td>1: Illegal access triggers due to a non-privileged write or read</td>
</tr>
<tr>
<td>This bit is set when an illegal access event occurs. This bit is cleared by writing 1 to it.</td>
<td></td>
</tr>
</tbody>
</table>
18.6.7 EXTI privilege configuration register ( EXTI_PRIVCFGFR1)

Address offset: 0x018
Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged read returns the register data.

Contains only register bits for privilege capable input events.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>SECx: Security enable on event input x (x = 31 to 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When EXTI_PRIVCFGFR.PRIVx is disabled, SECx can be accessed with privileged and unprivileged access.</td>
</tr>
<tr>
<td></td>
<td>When EXTI_PRIVCFGFR.PRIVx is enabled, SECx can only be written with privileged access.</td>
</tr>
<tr>
<td></td>
<td>Unprivileged write to this SECx is discarded.</td>
</tr>
<tr>
<td></td>
<td>0: Event security disabled (non-secure)</td>
</tr>
<tr>
<td></td>
<td>1: Event security enabled (secure)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>PRIVx: Security enable on event input x (x = 31 to 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When EXTI_SECCFGFR.SECx is disabled, PRIVx can be accessed with secure and non-secure access.</td>
</tr>
<tr>
<td></td>
<td>When EXTI_SECCFGFR.SECx is enabled, PRIVx can only be written with secure access.</td>
</tr>
<tr>
<td></td>
<td>Non-secure write to this PRIVx is discarded.</td>
</tr>
<tr>
<td></td>
<td>0: Event privilege disabled (unprivileged)</td>
</tr>
<tr>
<td></td>
<td>1: Event privilege enabled (privileged)</td>
</tr>
</tbody>
</table>

18.6.8 EXTI rising trigger selection register 2 ( EXTI_RTSR2)

Address offset: 0x020
Reset value: 0x0000 0000

Contains only register bits for configurable events.

| Bits 31:22, 20:19, 17:15, 13:0  | Reserved, must be kept at reset value. |

720/3119  RM0481 Rev 1
18.6.9 EXTI falling trigger selection register 2 (EXTI_FTSR2)

Address offset: 0x024

Reset value: 0x0000 0000

Contains only register bits for configurable events.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT46</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.

    0: Rising trigger disabled (for event and interrupt) for input line
    1: Rising trigger enabled (for event and interrupt) for input line

   1.

    Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.
Bits 31:22, 20:19, 17:15, 13:0 Reserved, must be kept at reset value.

Bit 21 **FT53**: Falling trigger event configuration bit of configurable event input x (1)
When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, FTx can only be accessed with secure access.
Non-secure write to this FTx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FTx can only be accessed with privileged access.
Unprivileged write to this FTx is discarded, unprivileged read returns 0.
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

Bit 18 **FT50**: Falling trigger event configuration bit of configurable event input x (1)
When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, FTx can only be accessed with secure access.
Non-secure write to this FTx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FTx can only be accessed with privileged access.
Unprivileged write to this FTx is discarded, unprivileged read returns 0.
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

Bit 14 **FT46**: Falling trigger event configuration bit of configurable event input x (1)
When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, FTx can only be accessed with secure access.
Non-secure write to this FTx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FTx can only be accessed with privileged access.
Unprivileged write to this FTx is discarded, unprivileged read returns 0.
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.
18.6.10 EXTI software interrupt event register 2 (EXTI_SWIER2)

Address offset: 0x028
Reset value: 0x0000 0000
Contains only register bits for configurable events.

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

Bits 31:22, 20:19, 17:15, 13:0 Reserved, must be kept at reset value.

Bit 21 **SWI53**: Software interrupt on event x
- When EXTI__SECCFGR__SECx is disabled, SWIx can be accessed with non-secure and secure access.
- When EXTI__SECCFGR__SECx is enabled, SWIx can only be accessed with secure access.
- Non-secure write to this SWIx is discarded, non-secure read returns 0.
- When EXTI__PRIVCFGR__PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.
- When EXTI__PRIVCFGR__PRIVx is enabled, SWIx can only be accessed with privileged access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.
- A software interrupt is generated independent from the setting in EXTI__RTSR and EXTI__FTSR. It always returns 0 when read.
  - 0: Writing 0 has no effect.
  - 1: Writing 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.

Bit 18 **SWI50**: Software interrupt on event x
- When EXTI__SECCFGR__SECx is disabled, SWIx can be accessed with non-secure and secure access.
- When EXTI__SECCFGR__SECx is enabled, SWIx can only be accessed with secure access.
- Non-secure write to this SWIx is discarded, non-secure read returns 0.
- When EXTI__PRIVCFGR__PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.
- When EXTI__PRIVCFGR__PRIVx is enabled, SWIx can only be accessed with privileged access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.
- A software interrupt is generated independent from the setting in EXTI__RTSR and EXTI__FTSR. It always returns 0 when read.
  - 0: Writing 0 has no effect.
  - 1: Writing 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.
18.6.11 EXTI rising edge pending register 2 (EXTI_RPR2)

Address offset: 0x02C
Reset value: 0x0000 0000
Contains only register bits for configurable events.

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

Bits 31:22, 20:19, 17:15, 13:0 Reserved, must be kept at reset value.

Bit 21 **RPIF53**: configurable event inputs x rising edge pending bit
When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, RPIFx can only be accessed with secure access.
Non-secure write to this RPIFx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can only be accessed with privileged access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.
A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.
0: No rising edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bit 14 **SWI46**: Software interrupt on event x
When EXTI_SECCFGR.SECx is disabled, SWIx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, SWIx can only be accessed with secure access.
Non-secure write to this SWIx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can only be accessed with privileged access. Unprivileged write to this SWIx is discarded, non-privileged read returns 0.
A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.
0: Writing 0 has no effect.
1: Writing 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.
Bit 18 **RPIF50**: configurable event inputs x rising edge pending bit

When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, RPIFx can only be accessed with secure access.
Non-secure write to this RPIFx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can only be accessed with privileged access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.
0: No rising edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bit 14 **RPIF46**: configurable event inputs x rising edge pending bit

When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, RPIFx can only be accessed with secure access.
Non-secure write to this RPIFx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can only be accessed with privileged access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.
0: No rising edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

### 18.6.12 EXTI falling edge pending register 2 (EXTI_FPR2)

Address offset: 0x030
Reset value: 0x0000 0000
Contains only register bits for configurable events.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPIF53</td>
<td>FPIF52</td>
<td>FPIF51</td>
<td>FPIF50</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FPIF48</td>
<td>FPIF47</td>
<td>FPIF46</td>
<td>FPIF45</td>
<td>FPIF44</td>
<td>FPIF43</td>
<td>FPIF42</td>
<td>FPIF41</td>
<td>FPIF40</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:22, 20:19, 17:15, 13:0 Reserved, must be kept at reset value.
Bit 21  **FPIF53**: configurable event inputs x falling edge pending bit
When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, FPIFx can only be accessed with secure access. Non-secure write to this FPIFx is discarded, non-secure read returns 0. When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can only be accessed with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.
0: No falling edge trigger request occurred
1: Falling edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bit 18  **FPIF50**: configurable event inputs x falling edge pending bit
When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, FPIFx can only be accessed with secure access. Non-secure write to this FPIFx is discarded, non-secure read returns 0. When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can only be accessed with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.
0: No falling edge trigger request occurred
1: Falling edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bit 14  **FPIF46**: configurable event inputs x falling edge pending bit
When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, FPIFx can only be accessed with secure access. Non-secure write to this FPIFx is discarded, non-secure read returns 0. When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can only be accessed with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.
0: No falling edge trigger request occurred
1: Falling edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.
18.6.13 EXTI security configuration register 2 (EXTI_SECCFGR2)

Address offset: 0x034
Reset value: 0x0000 0000

This register provides write access security, a non-secure write access is ignored and causes the generation of an illegal access event. A non-secure read returns the register data.

Contains only register bits for privilege capable input events.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEC57</td>
<td>SEC56</td>
<td>SEC55</td>
<td>SEC54</td>
<td>SEC53</td>
<td>SEC52</td>
<td>SEC51</td>
<td>SEC50</td>
<td>SEC49</td>
<td>SEC48</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:26 Reserved, must be kept at reset value.
Bits 25:0 **SECx**: Security enable on event input x (x = 57 to 32)
- When EXTI_PRIVCFGR.PRIVx is disabled, SECx can be accessed with privileged and unprivileged access.
- When EXTI_PRIVCFGR.PRIVx is enabled, SECx can only be written with privileged access.
- Unprivileged write to this SECx is discarded.
- 0: Event security disabled (non-secure)
- 1: Event security enabled (secure)

18.6.14 EXTI privilege configuration register 2 (EXTI_PRIVCFGR2)

Address offset: 0x038
Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged read returns the register data.

Contains only register bits for security capable input events.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PRIV57</td>
<td>PRIV56</td>
<td>PRIV55</td>
<td>PRIV54</td>
<td>PRIV53</td>
<td>PRIV52</td>
<td>PRIV51</td>
<td>PRIV50</td>
<td>PRIV49</td>
<td>PRIV48</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

PRIV47| PRIV46| PRIV45| PRIV44| PRIV43| PRIV42| PRIV41| PRIV40| PRIV39| PRIV38| PRIV37| PRIV36| PRIV35| PRIV34| PRIV33| PRIV32
| rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  |    |

Bits 31:26 Reserved, must be kept at reset value.
**18.6.15 EXTI external interrupt selection register (EXTI_EXTICR1)**

Address offset: 0x060 EXTI mux 0, 1, 2, 3

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:24</td>
<td>EXTI[7:0]: EXTI3 GPIO port selection</td>
</tr>
</tbody>
</table>

These bits are written by software to select the source input for EXTI3 external interrupt.

- When EXTI_SECCFGR1.SEC3 is disabled, EXTI3 can be accessed with non-secure and secure access.
- When EXTI_SECCFGR1.SEC3 is enabled, EXTI3 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
- When EXTI_PRIVCFGR1.PRIV3 is disabled, EXTI3 can be accessed with privileged and unprivileged access.
- When EXTI_PRIVCFGR1.PRIV3 is enabled, EXTI3 can only be accessed with privileged access. Unprivileged write to this bit is discarded.

- 0x00: PA3 pin
- 0x01: PB3 pin
- 0x02: PC3 pin
- 0x03: PD3 pin
- 0x04: PE3 pin
- 0x05: PF3 pin
- 0x06: PG3 pin
- 0x07: PH3 pin
- 0x08: PI3 pin
- Others: reserved
Bits 23:16 **EXTI2[7:0]**: EXTI2 GPIO port selection

These bits are written by software to select the source input for EXTI2 external interrupt.
When EXTI_SECCFG1.SEC2 is disabled, EXTI2 can be accessed with non-secure and secure access.
When EXTI_SECCFG1.SEC2 is enabled, EXTI2 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFG1.PRIV2 is disabled, EXTI2 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFG1.PRIV2 is enabled, EXTI2 can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0x00: PA2 pin  
0x01: PB2 pin  
0x02: PC2 pin  
0x03: PD2 pin  
0x04: PE2 pin  
0x05: PF2 pin  
0x06: PG2 pin  
0x07: PH2 pin  
0x08: PI2 pin  
Others: reserved

Bits 15:8 **EXTI1[7:0]**: EXTI1 GPIO port selection

These bits are written by software to select the source input for EXTI1 external interrupt.
When EXTI_SECCFG1.SEC1 is disabled, EXTI1 can be accessed with non-secure and secure access.
When EXTI_SECCFG1.SEC1 is enabled, EXTI1 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFG1.PRIV1 is disabled, EXTI1 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFG1.PRIV1 is enabled, EXTI1 can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0x00: PA1 pin  
0x01: PB1 pin  
0x02: PC1 pin  
0x03: PD1 pin  
0x04: PE1 pin  
0x05: PF1 pin  
0x06: PG1 pin  
0x07: PH1 pin  
0x08: PI1 pin  
Others: reserved
Bits 7:0 EXTIO[7:0]: EXTIO GPIO port selection
These bits are written by software to select the source input for EXTIO external interrupt.
When EXTIO_SECCFG1.SECC0 is disabled, EXTIO can be accessed with non-secure and secure access.
When EXTIO_SECCFG1.SECC0 is enabled, EXTIO can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTIO_PRIVCFG1.PRIV0 is disabled, EXTIO can be accessed with privileged and unprivileged access.
When EXTIO_PRIVCFG1.PRIV0 is enabled, EXTIO can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA0 pin
0x01: PB0 pin
0x02: PC0 pin
0x03: PD0 pin
0x04: PE0 pin
0x05: PF0 pin
0x06: PG0 pin
0x07: PH0 pin
0x08: PI0 pin
Others: reserved

18.6.16 EXTI external interrupt selection register (EXTI_EXTICR2)
Address offset: 0x064 EXTI mux 4, 5, 6, 7
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>EXTI[7:0]</th>
<th>EXTI[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>EXTI[7:0]</td>
<td>EXTI[7:0]</td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:24 EXT17[7:0]: EXTI7 GPIO port selection

These bits are written by software to select the source input for EXTI7 external interrupt.
When EXTI_SECCFG1.SECC7 is disabled, EXTI7 can be accessed with non-secure and secure access.
When EXTI_SECCFG1.SECC7 is enabled, EXTI7 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFG1.PRIV7 is disabled, EXTI7 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFG1.PRIV7 is enabled, EXTI7 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA7 pin
0x01: PB7 pin
0x02: PC7 pin
0x03: PD7 pin
0x04: PE7 pin
0x05: PF7 pin
0x06: PG7 pin
0x07: PH7 pin
0x08: PI7 pin
Others: reserved

Bits 23:16 EXT16[7:0]: EXTI6 GPIO port selection

These bits are written by software to select the source input for EXTI6 external interrupt.
When EXTI_SECCFG1.SECC6 is disabled, EXTI6 can be accessed with non-secure and secure access.
When EXTI_SECCFG1.SECC6 is enabled, EXTI6 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFG1.PRIV6 is disabled, EXTI6 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFG1.PRIV6 is enabled, EXTI6 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA6 pin
0x01: PB6 pin
0x02: PC6 pin
0x03: PD6 pin
0x04: PE6 pin
0x05: PF6 pin
0x06: PG6 pin
0x07: PH6 pin
0x08: PI6 pin
Others: reserved
Bits 15:8  **EXTI5[7:0]: EXTI5 GPIO port selection**

These bits are written by software to select the source input for EXTI5 external interrupt.

- When EXTI_SECCFGR1.SEC5 is disabled, EXTI5 can be accessed with non-secure and secure access.
- When EXTI_SECCFGR1.SEC5 is enabled, EXTI5 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
- When EXTI_PRIVCFGR1.PRIV5 is disabled, EXTI5 can be accessed with privileged and unprivileged access.
- When EXTI_PRIVCFGR1.PRIV5 is enabled, EXTI5 can only be accessed with privileged access. Unprivileged write to this bit is discarded.

- 0x00: PA5 pin
- 0x01: PB5 pin
- 0x02: PC5 pin
- 0x03: PD5 pin
- 0x04: PE5 pin
- 0x05: PF5 pin
- 0x06: PG5 pin
- 0x07: PH5 pin
- 0x08: PI5 pin
- Others: reserved

Bits 7:0  **EXTI4[7:0]: EXTI4 GPIO port selection**

These bits are written by software to select the source input for EXTI4 external interrupt.

- When EXTI_SECCFGR1.SEC4 is disabled, EXTI4 can be accessed with non-secure and secure access.
- When EXTI_SECCFGR1.SEC4 is enabled, EXTI4 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
- When EXTI_PRIVCFGR1.PRIV4 is disabled, EXTI4 can be accessed with privileged and unprivileged access.
- When EXTI_PRIVCFGR1.PRIV4 is enabled, EXTI4 can only be accessed with privileged access. Unprivileged write to this bit is discarded.

- 0x00: PA4 pin
- 0x01: PB4 pin
- 0x02: PC4 pin
- 0x03: PD4 pin
- 0x04: PE4 pin
- 0x05: PF4 pin
- 0x06: PG4 pin
- 0x07: PH4 pin
- 0x08: PI4 pin
- Others: reserved
## 18.6.17 EXTI external interrupt selection register (EXTI_EXTICR3)

Address offset: 0x068 EXTI mux 8, 9, 10, 11

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>EXTI11[7:0]</th>
<th>EXTI10[7:0]</th>
<th>EXTI9[7:0]</th>
<th>EXTI8[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:24 **EXTI11[7:0]**: EXTI11 GPIO port selection

These bits are written by software to select the source input for EXTI11 external interrupt.

- When EXTI_SECCFGR1.SEC11 is disabled, EXTI11 can be accessed with non-secure and secure access.
- When EXTI_SECCFGR1.SEC11 is enabled, EXTI11 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
- When EXTI_PRIVCFGR1.PRIV11 is disabled, EXTI11 can be accessed with privileged and unprivileged access.
- When EXTI_PRIVCFGR1.PRIV11 is enabled, EXTI11 can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0x00: PA11 pin
0x01: PB11 pin
0x02: PC11 pin
0x03: PD11 pin
0x04: PE11 pin
0x05: PF11 pin
0x06: PG11 pin
0x07: PH11 pin
0x08: PI11 pin
Others: reserved
Bits 23:16  **EXTI10[7:0]**: EXTI10 GPIO port selection

These bits are written by software to select the source input for EXTI10 external interrupt.

When EXTI_SECCFG1.SEC10 is disabled, EXTI10 can be accessed with non-secure and secure access.

When EXTI_SECCFG1.SEC10 is enabled, EXTI10 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.

When EXTI_PRIVCFG1.PRIV10 is disabled, EXTI10 can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFG1.PRIV10 is enabled, EXTI10 can only be accessed with privileged access. Unprivileged write to this bit is discarded.

- 0x00: PA10 pin
- 0x01: PB10 pin
- 0x02: PC10 pin
- 0x03: PD10 pin
- 0x04: PE10 pin
- 0x05: PF10 pin
- 0x06: PG10 pin
- 0x07: PH10 pin
- 0x08: PI10 pin
- Others: reserved

Bits 15:8  **EXTI9[7:0]**: EXTI9 GPIO port selection

These bits are written by software to select the source input for EXTI9 external interrupt.

When EXTI_SECCFG1.SEC9 is disabled, EXTI9 can be accessed with non-secure and secure access.

When EXTI_SECCFG1.SEC9 is enabled, EXTI9 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.

When EXTI_PRIVCFG1.PRIV9 is disabled, EXTI9 can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFG1.PRIV9 is enabled, EXTI9 can only be accessed with privileged access. Unprivileged write to this bit is discarded.

- 0x00: PA9 pin
- 0x01: PB9 pin
- 0x02: PC9 pin
- 0x03: PD9 pin
- 0x04: PE9 pin
- 0x05: PF9 pin
- 0x06: PG9 pin
- 0x07: PH9 pin
- 0x08: PI9 pin
- Others: reserved
**Bits 7:0 EXTI8[7:0]: EXTI8 GPIO port selection**

These bits are written by software to select the source input for EXTI8 external interrupt.

- When EXTI_SECCFGR1.SEC8 is disabled, EXTI8 can be accessed with non-secure and secure access.
- When EXTI_SECCFGR1.SEC8 is enabled, EXTI8 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
- When EXTI_PRIVCFGR1.PRIV8 is disabled, EXTI8 can be accessed with privileged and unprivileged access.
- When EXTI_PRIVCFGR1.PRIV8 is enabled, EXTI8 can only be accessed with privileged access. Unprivileged write to this bit is discarded.

- 0x00: PA8 pin
- 0x01: PB8 pin
- 0x02: PC8 pin
- 0x03: PD8 pin
- 0x04: PE8 pin
- 0x05: PF8 pin
- 0x06: PG8 pin
- 0x07: PH8 pin
- 0x08: PI8 pin
- Others: reserved

### 18.6.18 EXTI external interrupt selection register ( EXTIEXTICR4)

Address offset: 0x060 EXTI mux 12, 13, 14, 15

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>EXTI15[7:0]</th>
<th>EXTI14[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
</tr>
<tr>
<td>28</td>
<td>27</td>
<td>26</td>
</tr>
<tr>
<td>25</td>
<td>24</td>
<td>23</td>
</tr>
<tr>
<td>22</td>
<td>21</td>
<td>20</td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXTI13[7:0]</th>
<th>EXTI12[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXTI11[7:0]</th>
<th>EXTI10[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXTI9[7:0]</th>
<th>EXTI8[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXTI7[7:0]</th>
<th>EXTI6[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXTI5[7:0]</th>
<th>EXTI4[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Bits 31:24  **EXTI15[7:0]**: EXTI15 GPIO port selection
These bits are written by software to select the source input for EXTI15 external interrupt. When EXTI_SECCFGR1.SEC15 is disabled, EXTI15 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC15 is enabled, EXTI15 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV15 is disabled, EXTI15 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV15 is enabled, EXTI15 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA15 pin
0x01: PB15 pin
0x02: PC15 pin
0x03: PD15 pin
0x04: PE15 pin
0x05: PF15 pin
0x06: PG15 pin
0x07: PH15 pin
Others: reserved

Bits 23:16  **EXTI14[7:0]**: EXTI14 GPIO port selection
These bits are written by software to select the source input for EXTI14 external interrupt. When EXTI_SECCFGR1.SEC14 is disabled, EXTI14 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC14 is enabled, EXTI14 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV14 is disabled, EXTI14 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV14 is enabled, EXTI14 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA14 pin
0x01: PB14 pin
0x02: PC14 pin
0x03: PD14 pin
0x04: PE14 pin
0x05: PF14 pin
0x06: PG14 pin
0x07: PH14 pin
Others: reserved
Bits 15:8 **EXTI13[7:0]**: EXTI13 GPIO port selection

These bits are written by software to select the source input for EXTI13 external interrupt. When EXTI_SECCFGR1.SEC13 is disabled, EXTI13 can be accessed with non-secure and secure access.

When EXTI_SECCFGR1.SEC13 is enabled, EXTI13 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.

When EXTI_PRIVCFGR1.PRIV13 is disabled, EXTI13 can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR1.PRIV13 is enabled, EXTI13 can only be accessed with privileged access. Unprivileged write to this bit is discarded.

- 0x00: PA13 pin
- 0x01: PB13 pin
- 0x02: PC13 pin
- 0x03: PD13 pin
- 0x04: PE13 pin
- 0x05: PF13 pin
- 0x06: PG13 pin
- 0x07: PH13 pin
- Others: reserved

Bits 7:0 **EXTI12[7:0]**: EXTI12 GPIO port selection

These bits are written by software to select the source input for EXTI12 external interrupt.

When EXTI_SECCFGR1.SEC12 is disabled, EXTI12 can be accessed with non-secure and secure access.

When EXTI_SECCFGR1.SEC12 is enabled, EXTI12 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.

When EXTI_PRIVCFGR1.PRIV12 is disabled, EXTI12 can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR1.PRIV12 is enabled, EXTI12 can only be accessed with privileged access. Unprivileged write to this bit is discarded.

- 0x00: PA12 pin
- 0x01: PB12 pin
- 0x02: PC12 pin
- 0x03: PD12 pin
- 0x04: PE12 pin
- 0x05: PF12 pin
- 0x06: PG12 pin
- 0x07: PH12 pin
- Others: reserved
18.6.19 EXTI lock register (EXTI_LOCKR)

Address offset: 0x070
Reset value: 0x0000 0000

This register provides write access security: a non-secure write access is ignored and a read access returns zero data, and both generates an illegal access event.

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

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

Bit 0 **LOCK**: Global security and privilege configuration registers (EXTI_SECCFGR and EXTI_PRIVCFGR) lock

- This bit is written once after reset.
- 0: Security and privilege configuration open, can be modified.
- 1: Security and privilege configuration locked, can no longer be modified.

18.6.20 EXTI CPU wakeup with interrupt mask register (EXTI_IMR1)

Address offset: 0x080
Reset value: 0xFFFE 0000

Contains register bits for configurable events.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>IM31</td>
<td>IM30</td>
<td>IM29</td>
<td>IM28</td>
<td>IM27</td>
<td>IM26</td>
<td>IM25</td>
<td>IM24</td>
<td>IM23</td>
<td>IM22</td>
<td>IM21</td>
<td>IM20</td>
<td>IM19</td>
<td>IM18</td>
<td>IM17</td>
<td>IM16</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>IM15</td>
<td>IM14</td>
<td>IM13</td>
<td>IM12</td>
<td>IM11</td>
<td>IM10</td>
<td>IM9</td>
<td>IM8</td>
<td>IM7</td>
<td>IM6</td>
<td>IM5</td>
<td>IM4</td>
<td>IM3</td>
<td>IM2</td>
<td>IM1</td>
<td>IM0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **IMx**: CPU wakeup with interrupt mask on event input x \(^{(1)}\) (x = 31 to 0)

- When EXTI_SECCFGR.SECx is disabled, IMx can be accessed with non-secure and secure access.
- When EXTI_SECCFGR.SECx is enabled, IMx can only be accessed with secure access. Non-secure write to this bit is discarded and non-secure read returns 0.
- When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privileged and unprivileged access.
- When EXTI_PRIVCFGR.PRIVx is enabled, IMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.

0: Wakeup with interrupt request from input event x is masked.
1: Wakeup with interrupt request from input event x is unmasked.

1. The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.
18.6.21 EXTI CPU wakeup with event mask register (EXTI_EMR1)

Address offset: 0x084
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>EM15</td>
<td>EM14</td>
<td>EM13</td>
<td>EM12</td>
<td>EM11</td>
<td>EM10</td>
<td>EM9</td>
<td>EM8</td>
<td>EM7</td>
<td>EM6</td>
<td>EM5</td>
<td>EM4</td>
<td>EM3</td>
<td>EM2</td>
<td>EM1</td>
<td>EM0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 EMx: CPU wakeup with event generation mask on event input x (x = 31 to 0)
When EXTI_SECCFGR.SECx is disabled, EMx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, EMx can only be accessed with secure access.
Non-secure write to this bit x is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, EMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0: Wakeup with event generation from Line x is masked.
1: Wakeup with event generation from Line x is unmasked.

18.6.22 EXTI CPU wakeup with interrupt mask register 2 (EXTI_IMR2)

Address offset: 0x090
Reset value: 0x03DB BFFF
Contains register bits for configurable events.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>IM57</td>
<td>IM56</td>
<td>IM55</td>
<td>IM54</td>
<td>IM53</td>
<td>IM52</td>
<td>IM51</td>
<td>IM50</td>
<td>IM49</td>
<td>IM48</td>
<td>IM47</td>
<td>IM46</td>
<td>IM45</td>
<td>IM44</td>
<td>IM43</td>
<td>IM42</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>IM47</td>
<td>IM46</td>
<td>IM45</td>
<td>IM44</td>
<td>IM43</td>
<td>IM42</td>
<td>IM41</td>
<td>IM40</td>
<td>IM39</td>
<td>IM38</td>
<td>IM37</td>
<td>IM36</td>
<td>IM35</td>
<td>IM34</td>
<td>IM33</td>
<td>IM32</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 25:0 IMx: CPU wake up with interrupt mask on event input x (1) (x = 57 to 0)
When EXTI_SECCFGR.SECx is disabled, IMx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, IMx can only be accessed with secure access.
Non-secure write to this bit x is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, IMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0: Wakeup with interrupt request from input event x is masked.
1: Wakeup with interrupt request from input event x is unmasked.
1. The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

**18.6.23 EXTI CPU wakeup with event mask register 2 (EXTI_EMR2)**

Address offset: 0x094

Reset value: 0x0000 0000

---

**Table 142. EXTI register map and reset values**

| Offset | Register      | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | EXTI_RTSR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x004  | EXTI_FTSR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008  | EXTI_SWIER1   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00C  | EXTI_RPR1     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x010  | EXTI_FPR1     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x014  | EXTI_SECCFGR1 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

---

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

Bits 25:0 **EMx**: CPU wakeup with event generation mask on event input x (x = 57 to 32)

When EXTI_SECCFGR.SECx is disabled, EMx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, EMx can only be accessed with secure access.
Non-secure write to this bit x is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, EMx can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0: Wakeup with event generation from Line x is masked.
1: Wakeup with event generation from Line x is unmasked.

**18.6.24 EXTI register map**

---
## Table 142. EXTI register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>31</th>
<th>24</th>
<th>19</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x018</td>
<td>EXTI_PRIVCFGR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x020</td>
<td>EXTI_RTSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x024</td>
<td>EXTI_FTSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x028</td>
<td>EXTI_SWIER2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x02C</td>
<td>EXTI_RPR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x030</td>
<td>EXTI_FPR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x034</td>
<td>EXTI_SECCFGR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x038</td>
<td>EXTI_PRIVCFGR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x03C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x03D</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x03E</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x03F</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x060</td>
<td>EXTI_EXTICR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x064</td>
<td>EXTI_EXTICR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x068</td>
<td>EXTI_EXTICR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x06C</td>
<td>EXTI_EXTICR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x070</td>
<td>EXTI_LOCKR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x074</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x080</td>
<td>EXTI_IMR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Refer to Section 2.3 for the register boundary addresses.
19 Cyclic redundancy check calculation unit (CRC)

19.1 Introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

19.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  \[X^{32} + X^{26} + X^{23} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + 1\]
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-, 16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes
19.3 CRC functional description

19.3.1 CRC block diagram

Figure 91. CRC calculation unit block diagram

19.3.2 CRC internal signals

Table 143. CRC internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crc_hclk</td>
<td>Digital</td>
<td>AHB clock</td>
</tr>
</tbody>
</table>

19.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit accesses are allowed.

The duration of the computation depends on data width:
- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written without waiting for any wait states due to the previous CRC calculation.
The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte
- 0xD458B23C with bit-reversal done by half-word
- 0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register.

The operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

**Polynomial programmability**

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

*Note:* The type of an even polynomial is $X+X^2+..+X^n$, while the type of an odd polynomial is $1+X+X^2+..+X^n$.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size cannot be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.
19.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes. For the other registers only 32-bit accesses are allowed.

19.4.1 CRC data register (CRC_DR)

Address offset: 0x00
Reset value: 0xFFFF FFFF

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

Bits 31:0 DR[31:0]: Data register bits
This register is used to write new data to the CRC calculator.
It holds the previous CRC calculation result when it is read.
If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

19.4.2 CRC independent data register (CRC_IDR)

Address offset: 0x04
Reset value: 0x0000 0000

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

Bits 31:0 IDR[31:0]: General-purpose 32-bit data register bits
These bits can be used as a temporary storage location for four bytes.
This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register.
19.4.3 CRC control register (CRC_CR)

Address offset: 0x08
Reset value: 0x0000 0000

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

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

Bit 7 REV_OUT: Reverse output data
- This bit controls the reversal of the bit order of the output data.
- 0: Bit order not affected
- 1: Bit-reversed output format

Bits 6:5 REV_IN[1:0]: Reverse input data
- This bitfield controls the reversal of the bit order of the input data
- 00: Bit order not affected
- 01: Bit reversal done by byte
- 10: Bit reversal done by half-word
- 11: Bit reversal done by word

Bits 4:3 POLYSIZE[1:0]: Polynomial size
- These bits control the size of the polynomial.
- 00: 32 bit polynomial
- 01: 16 bit polynomial
- 10: 8 bit polynomial
- 11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 RESET: RESET bit
- This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware.
19.4.4 CRC initial value (CRC_INIT)

Address offset: 0x10
Reset value: 0xFFFF FFFF

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

Bits 31:0 CRC_INIT[31:0]: Programmable initial CRC value
This register is used to write the CRC initial value.

19.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14
Reset value: 0x04C1 1DB7

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

Bits 31:0 POL[31:0]: Programmable polynomial
This register is used to write the coefficients of the polynomial to be used for CRC calculation.
If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.
### 19.4.6 CRC register map

Table 144. CRC register map and reset values

| Offset | Register name | Offset Name | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x00   | CRC_DR        |             | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   |     |     |     |
|        | Reset value   |             | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   |     |     |     |
| 0x04   | CRC_IDR       |             | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |     |     |     |
|        | Reset value   |             | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |     |     |     |
| 0x08   | CRC_CR        |             | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |     |     |     |
|        | Reset value   |             | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |     |     |     |
| 0x10   | CRC_INIT      |             | 0   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   |     |     |     |
|        | Reset value   |             | 0   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   |     |     |     |
| 0x14   | CRC_POL       |             | 0   | 0   | 0   | 0   | 0   | 1   | 0   | 0   | 1   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 1   | 0   | 0   | 0   | 1   | 1   | 0   | 1   | 0   | 1   | 1   |     |     |     |
|        | Reset value   |             | 0   | 0   | 0   | 0   | 0   | 1   | 0   | 0   | 1   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 1   | 0   | 0   | 0   | 1   | 1   | 0   | 1   | 0   | 1   | 1   |     |     |     |

Refer to Section 2.3 on page 111 for the register boundary addresses.
20 CORDIC co-processor (CORDIC)

20.1 CORDIC introduction

The CORDIC co-processor provides hardware acceleration of mathematical functions (mainly trigonometric ones) commonly used in motor control, metering, signal processing and many other applications.

It speeds up the calculation of these functions compared to a software implementation, making it possible the use of a lower operating frequency, or freeing up processor cycles in order to perform other tasks.

20.2 CORDIC main features

- 24-bit CORDIC rotation engine
- Circular and Hyperbolic modes
- Rotation and Vectoring modes
- Functions: sine, cosine, sinh, cosh, atan, atan2, atanh, modulus, square root, natural logarithm
- Programmable precision
- Low latency AHB slave interface
- Results can be read as soon as ready, without polling or interrupt
- DMA read and write channels
- Multiple register read/write by DMA

20.3 CORDIC functional description

20.3.1 General description

The CORDIC is a cost-efficient successive approximation algorithm for evaluating trigonometric and hyperbolic functions.

In trigonometric (circular) mode, the sine and cosine of an angle $\theta$ are determined by rotating the unit vector $[1, 0]$ through decreasing angles until the cumulative sum of the rotation angles equals the input angle $\theta$. The $x$ and $y$ cartesian components of the rotated vector then correspond, respectively, to the cosine and sine of $\theta$. Inversely, the angle of a vector $[x, y]$ corresponding to arctangent ($y / x$), is determined by rotating $[x, y]$ through successively decreasing angles to obtain the unit vector $[1, 0]$. The cumulative sum of the rotation angles gives the angle of the original vector.

The CORDIC algorithm can also be used for calculating hyperbolic functions ($\sinh$, $\cosh$, $\text{atanh}$), by replacing the successive circular rotations by steps along a hyperbole.

Other functions can be derived from the basic functions described above.

20.3.2 CORDIC functions

The first step when using the co-processor is to select the required function, by programming the FUNC field of the CORDIC_CR register accordingly.
Table 145 lists the functions supported by the CORDIC co-processor.

### Table 145. CORDIC functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Primary argument (ARG1)</th>
<th>Secondary argument (ARG2)</th>
<th>Primary result (RES1)</th>
<th>Secondary result (RES2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cosine</td>
<td>angle ( \theta )</td>
<td>modulus ( m )</td>
<td>( m \cdot \cos \theta )</td>
<td>( m \cdot \sin \theta )</td>
</tr>
<tr>
<td>Sine</td>
<td>angle ( \theta )</td>
<td>modulus ( m )</td>
<td>( m \cdot \sin \theta )</td>
<td>( m \cdot \cos \theta )</td>
</tr>
<tr>
<td>Phase</td>
<td>( x )</td>
<td>( y )</td>
<td>( \arctan(2)(y,x) )</td>
<td>( \sqrt{x^2 + y^2} )</td>
</tr>
<tr>
<td>Modulus</td>
<td>( x )</td>
<td>( y )</td>
<td>( \sqrt{x^2 + y^2} )</td>
<td>( \arctan(2)(y,x) )</td>
</tr>
<tr>
<td>Arctangent</td>
<td>( x )</td>
<td>none</td>
<td>( \tan^{-1} x )</td>
<td>none</td>
</tr>
<tr>
<td>Hyperbolic cosine</td>
<td>( x )</td>
<td>none</td>
<td>( \cosh x )</td>
<td>( \sinh x )</td>
</tr>
<tr>
<td>Hyperbolic sine</td>
<td>( x )</td>
<td>none</td>
<td>( \sinh x )</td>
<td>( \cosh x )</td>
</tr>
<tr>
<td>Hyperbolic arctangent</td>
<td>( x )</td>
<td>none</td>
<td>( \tanh^{-1} x )</td>
<td>none</td>
</tr>
<tr>
<td>Natural logarithm</td>
<td>( x )</td>
<td>none</td>
<td>( \ln x )</td>
<td>none</td>
</tr>
<tr>
<td>Square root</td>
<td>( x )</td>
<td>none</td>
<td>( \sqrt{x} )</td>
<td>none</td>
</tr>
</tbody>
</table>

Several functions take two input arguments (ARG1 and ARG2) and some generate two results (RES1 and RES2) simultaneously. This is a side-effect of the algorithm and means that only one operation is needed to obtain two values. This is the case, for example, when performing polar-to-rectangular conversion: \( \sin \theta \) also generates \( \cos \theta \), \( \cos \theta \) also generates \( \sin \theta \). Similarly for rectangular-to-polar conversion (phase\((x,y)\), modulus\((x,y)\)) and for hyperbolic functions (\( \cosh \theta , \sinh \theta \)).

**Note:** The exponential function, \( \exp x \), can be obtained as the sum of \( \sinh x \) and \( \cosh x \). Furthermore, base \( N \) logarithms, \( \log_N x \), can be derived by multiplying \( \ln x \) by a constant \( K \), where \( K = 1/\ln N \).

For certain functions (atan, log, sqrt) a scaling factor (see Section 20.3.4) can be applied to extend the range of the function beyond the maximum \([-1, 1]\) supported by the q1.31 fixed point format. The scaling factor must be set to 0 for all other circular functions, and to 1 for hyperbolic functions.

### Cosine

#### Table 146. Cosine parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>Angle ( \theta ) in radians, divided by ( \pi )</td>
<td>([-1, 1])</td>
</tr>
<tr>
<td>ARG2</td>
<td>Modulus ( m )</td>
<td>([0, 1])</td>
</tr>
<tr>
<td>RES1</td>
<td>( m \cdot \cos \theta )</td>
<td>([-1, 1])</td>
</tr>
</tbody>
</table>
This function calculates the cosine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the cosine of the angle, multiplied by the modulus.

The secondary result, RES2, is the sine of the angle, multiplied by the modulus.

Sine

This function calculates the sine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the sine of the angle, multiplied by the modulus.

The secondary result, RES2, is the cosine of the angle, multiplied by the modulus.

Phase

This function calculates the cosine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the cosine of the angle, multiplied by the modulus.

The secondary result, RES2, is the cosine of the angle, multiplied by the modulus.
This function calculates the phase angle in the range \(-\pi\) to \(\pi\) of a vector \(v = [x \ y]\) (also known as \(\text{atan2}(y;x)\)). It can also be used to perform rectangular to polar conversion.

The primary argument is the \(x\) coordinate, that is, the magnitude of the vector in the direction of the \(x\) axis. If \(|x| > 1\), a scaling must be applied in software to adapt it to the \(q1.31\) range of \(\text{ARG1}\).

The secondary argument is the \(y\) coordinate, that is, the magnitude of the vector in the direction of the \(y\) axis. If \(|y| > 1\), a scaling must be applied in software to adapt it to the \(q1.31\) range of \(\text{ARG2}\).

The primary result, \(\text{RES1}\), is the phase angle \(\theta\) of the vector \(v\). \(\text{RES1}\) must be multiplied by \(\pi\) to obtain the angle in radians. Note that values close to \(\pi\) may sometimes wrap to \(-\pi\) due to the circular nature of the phase angle.

The secondary result, \(\text{RES2}\), is the modulus, given by: \(|v| = \sqrt{x^2 + y^2}\). If \(|v| > 1\) the result in \(\text{RES2}\) is saturated to 1.

**Modulus**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>(x) coordinate</td>
<td>([-1, 1])</td>
</tr>
<tr>
<td>ARG2</td>
<td>(y) coordinate</td>
<td>([-1, 1])</td>
</tr>
<tr>
<td>RES1</td>
<td>Modulus (m)</td>
<td>([0, 1])</td>
</tr>
<tr>
<td>RES2</td>
<td>Phase angle (\theta)</td>
<td>([-1, 1])</td>
</tr>
<tr>
<td>SCALE</td>
<td>Not applicable</td>
<td>0</td>
</tr>
</tbody>
</table>

This function calculates the magnitude, or modulus, of a vector \(v = [x \ y]\). It can also be used to perform rectangular to polar conversion.

The primary argument is the \(x\) coordinate, that is, the magnitude of the vector in the direction of the \(x\) axis. If \(|x| > 1\), a scaling must be applied in software to adapt it to the \(q1.31\) range of \(\text{ARG1}\).

The secondary argument is the \(y\) coordinate, that is, the magnitude of the vector in the direction of the \(y\) axis. If \(|y| > 1\), a scaling must be applied in software to adapt it to the \(q1.31\) range of \(\text{ARG2}\).

The primary result, \(\text{RES1}\), is the modulus, given by: \(|v| = \sqrt{x^2 + y^2}\). If \(|v| > 1\) the result in \(\text{RES1}\) is saturated to 1.

The secondary result, \(\text{RES2}\), is the phase angle \(\theta\) of the vector \(v\). \(\text{RES2}\) must be multiplied by \(\pi\) to obtain the angle in radians. Note that values close to \(\pi\) may sometimes wrap to \(-\pi\) due to the circular nature of the phase angle.
Arctangent

This function calculates the arctangent, or inverse tangent, of the input argument \( x \).

The primary argument, ARG1, is the input value, \( x = \tan \theta \). If \(|x| > 1\), a scaling factor of \( 2^{-n} \) must be applied in software such that \(-1 < x \cdot 2^{-n} < 1\). The scaled value \( x \cdot 2^{-n} \) is programmed in ARG1 and the scale factor \( n \) must be programmed in the SCALE parameter.

Note that the maximum input value allowed is \( \tan \theta = 128 \), which corresponds to an angle \( \theta = 89.55 \) degrees. For \(|x| > 128\), a software method must be used to find \( \tan^{-1} x \).

The secondary argument, ARG2, is unused.

The primary result, RES1, is the angle \( \theta = \tan^{-1} x \). RES1 must be multiplied by \( 2^n \cdot \pi \) to obtain the angle in radians.

The secondary result, RES2, is unused.

Hyperbolic cosine

This function calculates the hyperbolic cosine of a hyperbolic angle \( x \). It can also be used to calculate the exponential functions \( e^x = \cosh x + \sinh x \), and \( e^{-x} = \cosh x - \sinh x \).

The primary argument is the hyperbolic angle \( x \). Only values of \( x \) in the range \(-1.118 \) to \(+1.118\) are supported. Since the minimum value of \( \cosh x \) is 1, which is beyond the range of the q1.31 format, a scaling factor of \( 2^n \) must be applied in software. The factor \( n = 1 \) must be programmed in the SCALE parameter.

The secondary argument is not used.

---

### Table 150. Arctangent parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>( x \cdot 2^{-n} )</td>
<td>([-1, 1])</td>
</tr>
<tr>
<td>ARG2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>RES1</td>
<td>( 2^n \cdot \tan^{-1} x ), in radians, divided by ( \pi )</td>
<td>([-1, 1])</td>
</tr>
<tr>
<td>RES2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>SCALE</td>
<td>( n )</td>
<td>([0, 7])</td>
</tr>
</tbody>
</table>

### Table 151. Hyperbolic cosine parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>( x \cdot 2^{-n} )</td>
<td>([-0.559, 0.559])</td>
</tr>
<tr>
<td>ARG2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>RES1</td>
<td>( 2^n \cdot \cosh x )</td>
<td>([0.5, 0.846])</td>
</tr>
<tr>
<td>RES2</td>
<td>( 2^n \cdot \sinh x )</td>
<td>([-0.683, 0.683])</td>
</tr>
<tr>
<td>SCALE</td>
<td>( n )</td>
<td>1</td>
</tr>
</tbody>
</table>
The primary result, RES1, is the hyperbolic cosine, \( \cosh x \). RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic sine, \( \sinh x \). RES2 must be multiplied by 2 to obtain the correct result.

**Hyperbolic sine**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>( x \cdot 2^{-n} )</td>
<td>([-0.559, 0.559])</td>
</tr>
<tr>
<td>ARG2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>RES1</td>
<td>( 2^{-n} \cdot \sinh x )</td>
<td>([-0.683, 0.683])</td>
</tr>
<tr>
<td>RES2</td>
<td>( 2^{-n} \cdot \cosh x )</td>
<td>([0.5, 0.846])</td>
</tr>
<tr>
<td>SCALE</td>
<td>( n )</td>
<td>1</td>
</tr>
</tbody>
</table>

This function calculates the hyperbolic sine of a hyperbolic angle \( x \). It can also be used to calculate the exponential functions \( e^x = \cosh x + \sinh x \), and \( e^{-x} = \cosh x - \sinh x \).

The primary argument is the hyperbolic angle \( x \). Only values of \( x \) in the range \(-1.118\) to \(+1.118\) are supported. For all input values, a scaling factor of \( 2^{-n} \) must be applied in software, where \( n = 1 \). The scaled value \( x \cdot 0.5 \) is programmed in ARG1 and the factor \( n = 1 \) must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic sine, \( \sinh x \). RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic cosine, \( \cosh x \). RES2 must be multiplied by 2 to obtain the correct result.

**Hyperbolic arctangent**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>( x \cdot 2^{-n} )</td>
<td>([-0.403, 0.403])</td>
</tr>
<tr>
<td>ARG2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>RES1</td>
<td>( 2^{-n} \cdot \text{atanh} x )</td>
<td>([-0.559, 0.559])</td>
</tr>
<tr>
<td>RES2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>SCALE</td>
<td>( n )</td>
<td>1</td>
</tr>
</tbody>
</table>

This function calculates the hyperbolic arctangent of the input argument \( x \).

The primary argument is the input value \( x \). Only values of \( x \) in the \(-0.806\) to \(+0.806\) range are supported. The value \( x \) must be scaled by a factor \( 2^{-n} \), where \( n = 1 \). The scaled value
x · 0.5 is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result is the hyperbolic arctangent, atanh x. RES1 must be multiplied by 2 to obtain the correct value.

The secondary result is not used.

**Natural logarithm**

<table>
<thead>
<tr>
<th>Table 154. Natural logarithm parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>ARG1</td>
</tr>
<tr>
<td>ARG2</td>
</tr>
<tr>
<td>RES1</td>
</tr>
<tr>
<td>RES2</td>
</tr>
<tr>
<td>SCALE</td>
</tr>
</tbody>
</table>

This function calculates the natural logarithm of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.107 to 9.35 are supported. The value x must be scaled by a factor $2^{-n}$, such that $x · 2^{-n} < 1 - 2^{-n}$. The scaled value $x · 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 155 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

<table>
<thead>
<tr>
<th>Table 155. Natural log scaling factors and corresponding ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

The secondary argument is not used.

The primary result is the natural logarithm, ln x. RES1 must be multiplied by $2^{(n+1)}$ to obtain the correct value.

The secondary result is not used.
**Square root**

This function calculates the square root of the input argument \( x \).

The primary argument is the input value \( x \). Only values of \( x \) in the range 0.027 to 2.34 are supported. The value \( x \) must be scaled by a factor \( 2^{-n} \), such that \( x \cdot 2^{-n} < (1 - 2^{-(n-2)}) \).

The scaled value \( x \cdot 2^{-n} \) is programmed in ARG1 and the factor \( n \) must be programmed in the SCALE parameter.

*Table 157* lists the valid scaling factors, \( n \), and the corresponding ranges of \( x \) and ARG1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>( x \cdot 2^{-n} )</td>
<td>[0.027 0.875]</td>
</tr>
<tr>
<td>ARG2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>RES1</td>
<td>( 2^{-n} \sqrt{x} )</td>
<td>[0.04 1]</td>
</tr>
<tr>
<td>RES2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>SCALE</td>
<td>( n )</td>
<td>[0 2]</td>
</tr>
</tbody>
</table>

The secondary argument is not used.

The primary result is the square root of \( x \). RES1 must be multiplied by \( 2^n \) to obtain the correct value.

The secondary result is not used.

### 20.3.3 Fixed point representation

The CORDIC operates in fixed point signed integer format. Input and output values can be either q1.31 or q1.15.

In q1.31 format, numbers are represented by one sign bit and 31 fractional bits (binary decimal places). The numeric range is therefore \(-1 \) (0x80000000) to \( 1 - 2^{-31} \) (0x7FFFFFFF).

In q1.15 format, the numeric range is \( 1 \) (0x8000) to \( 1 - 2^{-15} \) (0x7FFF). This format has the advantage that two input arguments can be packed into a single 32-bit write, and two results can be fetched in one 32-bit read.

### 20.3.4 Scaling factor

Several of the functions listed in *Section 20.3.2* specify a scaling factor, SCALE. This allows the function input range to be extended to cover the full range of values supported by the CORDIC, without saturating the input, output or internal registers. If the scaling factor is
required, it has to be calculated in software and programmed into the SCALE field of the CORDIC_CSR register. The input arguments must be scaled accordingly before programming the scaled values in the CORDIC_WDATA register. The scaling must also be undone on the results read from the CORDIC_RDATA register.

*Note:* The scaling factor entails a loss of precision due to truncation of the scaled value.

### 20.3.5 Precision

The precision of the result is dependent on the number of CORDIC iterations. The algorithm converges at a constant rate of one binary digit per iteration for trigonometric functions (sine, cosine, phase, modulus), see Figure 92.

For hyperbolic functions (hyperbolic sine, hyperbolic cosine, natural logarithm), the convergence rate is less constant due to the peculiarities of the CORDIC algorithm (see Figure 93). The square root function converges at roughly twice the speed of the hyperbolic functions (see Figure 94).

![Figure 92. CORDIC convergence for trigonometric functions](image-url)
Figure 93. CORDIC convergence for hyperbolic functions

Hyperbolic Sine/Cosine Convergence of q1.23 fixed point Cordic

Max absolute error

Iterations

$10^{-1}$ $10^{-2}$ $10^{-3}$ $10^{-4}$ $10^{-5}$ $10^{-6}$ $10^{-7}$ $10^{-8}$ $10^{-9}$ $10^{-10}$

q1.15
q1.31
Figure 94. CORDIC convergence for square root

Note: The convergence rate decreases as the quantization error starts to become significant.

The CORDIC can perform four iterations per clock cycle. For each function, the maximum error remaining after every four iterations is shown in Table 158, together with the number of clock cycles required to reach that precision. From this table, the desired number of cycles can be determined and programmed in the PRECISION field of the CORDIC_CR register. The co-processor stops as soon as the programmed number of iterations has completed, and the result can be read immediately.

Table 158. Precision vs. number of iterations

<table>
<thead>
<tr>
<th>Function</th>
<th>Number of iterations</th>
<th>Number of cycles</th>
<th>Max residual error(^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>q1.31 format</td>
</tr>
<tr>
<td>Sin, Cos, Phase(^{(2)}), Mod, Atan(^{(4)})</td>
<td>4</td>
<td>1</td>
<td>(2^{-3})</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2</td>
<td>(2^{-7})</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>3</td>
<td>(2^{-11})</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>4</td>
<td>(2^{-15})</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>5</td>
<td>(2^{-18})</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>6</td>
<td>(2^{-19})</td>
</tr>
</tbody>
</table>
20.3.6 Zero-overhead mode

The fastest way to use the co-processor is to pre-program the CORDIC_CSR register with the function to be performed (FUNC), the desired number of clock cycles (PRECISION), the size of the input and output values (ARGSIZE, RESSIZE), the number of input arguments (NARGS) and/or results (NRES), and the scaling factor (SCALE), if applicable.

Subsequently, a calculation is triggered by writing the input arguments to the CORDIC_WDATA register. As soon as the correct number of input arguments has been written (and any ongoing calculation has finished) a new calculation is launched using these input arguments and the current CORDIC_CSR settings. There is no need to re-program the CORDIC_CSR register if there is no change.

If a dual 32-bit input argument is needed (ARGSIZE = 0, NARGS = 1), the primary input argument, ARG1, must be written first, followed by the secondary argument, ARG2. If the secondary argument remains unchanged for a series of calculations, the second write can be avoided, by re-programming the number of arguments to one (NARGS = 0), once the first calculation has started. The secondary argument retains its programmed value as long as the function is not changed.

Note: ARG2 is set to +1 (0x7FFFFFFF) after a reset.

If two 16-bit arguments are used (ARGSIZE = 1) they must be packed into a 32-bit word, with ARG1 in the least significant half-word and ARG2 in the most significant half-word. The packed 32-bit word is then written to the CORDIC_WDATA register. Only one write is needed in this case (NARGS = 0).

For functions taking only one input argument, ARG1, it is recommended to set NARGS = 0. If NARGS = 1, a second write to CORDIC_WDATA must be performed to trigger the calculation. The ARG2 data in this case is not used.
Once the calculation starts, any attempt to read the CORDIC_RDATA register inserts bus wait states until the calculation is completed, before returning the result. Hence it is possible for the software to write the input and immediately read the result without polling to see if it is valid. Alternatively, the processor can wait for the appropriate number of clock cycles before reading the result. This time can be used to program the CORDIC_CSR register for the next calculation and prepare the next input data, if needed. The CORDIC_CSR register can be re-programmed while a calculation is in progress, without affecting the result of the ongoing calculation. In the same way, the CORDIC_WDATA register can be updated with the next argument(s) once the previous arguments have been taken into account. The next arguments and settings remain pending until the previous calculation has completed.

When a calculation is finished, the result(s) can be read from the CORDIC_RDATA register. If two 32-bit results are expected (NRES = 1, RESSIZE = 0), the primary result (RES1) is read out first, followed by the secondary result (RES2). If only one 32-bit result is expected (NRES = 0, RESSIZE = 0), then RES1 is output on the first read.

If 16-bit results are expected (RESSIZE = 1), a single read to CORDIC_RDATA fetches both results packed into a 32-bit word. RES1 is in the lower half-word, and RES2 in the upper half-word. In this case, it is recommended to program NRES = 0. If NRES = 1, a second read of CORDIC_RDATA must be performed in order to free up the CORDIC for the next operation. The data from this second read must be discarded.

The next calculation starts when the expected number of results has been read, provided the expected number of arguments has been written. This means that at any time, there can be one calculation in progress, or waiting for the results to be read, and one operation pending. Any further access to CORDIC_WDATA while an operation is pending, cancels the pending operation and overwrites the data.

The following sequence summarizes the use of the CORDIC_IP in zero-overhead mode:

1. Program the CORDIC_CSR register with the appropriate settings
2. Program the argument(s) for the first calculation in the CORDIC_WDATA register. This launches the first calculation.
3. If needed, update the CORDIC_CSR register settings for the next calculation.
4. Program the argument(s) for the next calculation in the CORDIC_WDATA register.
5. Read the result(s) from the CORDIC_RDATA register. This triggers the next calculation.
6. Go to step 3.

### 20.3.7 Polling mode

When a new result is available in the CORDIC_RDATA register, the RRDY flag is set in the CORDIC_CSR register. The flag can be polled by reading the register. It is reset by reading the CORDIC_RDATA register (once or twice depending on the NRES field of the CORDIC_CSR register).

Polling the RRDY flag takes slightly longer than reading the CORDIC_RDATA register directly, since the result is not read as soon as it is available. However the processor and bus interface are not stalled while reading the CORDIC_CSR register, so this mode may be of interest if stalling the processor is not acceptable (e.g. if low latency interrupts must be serviced).
20.3.8 Interrupt mode

By setting the interrupt enable (IE) bit in the CORDIC_CSR register, an interrupt is generated whenever the RRDY flag is set. The interrupt is cleared when the flag is reset.

This mode allows the result of the calculation to be read under interrupt service routine, and hence given a priority relative to other tasks. However it is slower than directly reading the result, or polling the flag, due to the interrupt handling delays.

20.3.9 DMA mode

If the DMA write enable (DMAWEN) bit is set in the CORDIC_CSR register, and no operation is pending, a DMA write channel request is generated. The DMA controller can transfer a primary input argument (ARG1) from memory into the CORDIC_WDATA register. Writing into the register deasserts the DMA request. If NARGS = 1 in the CORDIC_CSR register, a second DMA write channel request is generated to transfer the secondary input argument (ARG2) into the CORDIC_WDATA register. When all input arguments have been written, and any ongoing calculation has been completed (by reading the results), a new calculation is started and another DMA write channel request is generated.

If the DMA read enable (DMAREN) bit is set in the CORDIC_CSR register, the RRDY flag going active generates a DMA read channel request. The DMA controller can then transfer the primary result (RES1) from the CORDIC_RDATA register to memory. Reading the register deasserts the DMA request. If NRES = 1 in the CORDIC_CSR register, a second DMA request is generated to read out the secondary result (RES2). When all results have been read, the RRDY flag is deasserted.

The DMA read and write channels can be enabled separately. If both channels are enabled, the CORDIC can autonomously perform repeated calculations on a buffer of data without processor intervention. This allows the processor to perform other tasks. The DMA controller is operating in memory-to-peripheral mode for the write channel, and peripheral-to-memory mode for the read channel. Note that the sequence is started by the processor setting the DMAWEN flag. Thereafter the DMA read and write requests are generated as fast as the CORDIC can process the data.

In some cases, the input data may be stored in memory, and the output is transferred at regular intervals to another peripheral, such as a digital-to-analog converter. In this case, the destination peripheral generates a DMA request each time it needs a new data. The DMA controller can directly fetch the next sample from the CORDIC_RDATA register (in this case the DMA controller is operating in memory-to-peripheral mode, even though the source is a peripheral register). The act of reading the result allows the CORDIC to start a new calculation, which in turn generates a DMA write channel request, and the DMA controller transfers the next input value to the CORDIC_WDATA register. The DMA write channel is enabled (DMAWEN = 1), but the read channel must not be enabled.

In a similar way, data coming from another peripheral, such as an ADC, can be transferred directly to the CORDIC_WDATA register (in peripheral-to-memory mode). The DMA write channel must not be enabled. The CORDIC processes the input data and generate a DMA read request when complete, if DMAREN = 1. The DMA controller then transfers the result from CORDIC_RDATA register to memory (peripheral-to-memory mode).

Note: No DMA request is generated to program the CORDIC_CSR register. DMA mode is therefore only useful when repeatedly performing the same function with the same settings. The scale factor cannot be changed during a series of DMA transfers.
Note: Each DMA request must be acknowledged, as a result of the DMA performing an access to the CORDIC_WDATA or CORDIC_RDATA register. If an extraneous access to the relevant register occurs before this, the acknowledge is asserted prematurely, and may block the DMA channel. Therefore, when the DMA read channel is enabled, CPU access to the CORDIC_RDATA register must be avoided. Similarly, the processor must avoid accessing the CORDIC_WDATA register when the DMA write channel is enabled.

20.4 CORDIC registers
The CORDIC registers can only be accessed in 32-bit word format

20.4.1 CORDIC control/status register (CORDIC_CSR)
Address offset: 0x00
Reset value: 0x0000 0050

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Reset Value</th>
<th>Access</th>
<th>Bit</th>
<th>Description</th>
<th>Reset Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RRDY: Result ready flag</td>
<td>0</td>
<td>r</td>
<td>30</td>
<td>Res: Result supported</td>
<td>0</td>
<td>r</td>
</tr>
<tr>
<td>29</td>
<td>Res: Result supported</td>
<td>0</td>
<td>r</td>
<td>28</td>
<td>Res: Result supported</td>
<td>0</td>
<td>r</td>
</tr>
<tr>
<td>27</td>
<td>Res: Result supported</td>
<td>0</td>
<td>r</td>
<td>26</td>
<td>Res: Result supported</td>
<td>0</td>
<td>r</td>
</tr>
<tr>
<td>25</td>
<td>Res: Result supported</td>
<td>0</td>
<td>r</td>
<td>24</td>
<td>Res: Result supported</td>
<td>0</td>
<td>r</td>
</tr>
<tr>
<td>23</td>
<td>Res: Result supported</td>
<td>0</td>
<td>r</td>
<td>22</td>
<td>Res: Result supported</td>
<td>0</td>
<td>r</td>
</tr>
<tr>
<td>21</td>
<td>ARG SIZE: Width of input data</td>
<td>0</td>
<td>rw</td>
<td>20</td>
<td>NARGS: Number of arguments</td>
<td>0</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>NARGS: Number of arguments</td>
<td>0</td>
<td>r</td>
<td>18</td>
<td>NRES DMA WEN DMA REN IEN</td>
<td>0</td>
<td>r</td>
</tr>
<tr>
<td>17</td>
<td>NRES DMA WEN DMA REN IEN</td>
<td>0</td>
<td>r</td>
<td>16</td>
<td>Res: Reserve supported</td>
<td>0</td>
<td>r</td>
</tr>
</tbody>
</table>

Bit 31 RRDY: Result ready flag
0: No new data in output register
1: CORDIC_RDATA register contains new data.

This bit is set by hardware when a CORDIC operation completes. It is reset by hardware when the CORDIC_RDATA register is read (NRES+1) times.
When this bit is set, if the IEN bit is also set, the CORDIC interrupt is asserted. If the DMAREN bit is set, a DMA read channel request is generated. While this bit is set, no new calculation is started.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 ARG SIZE: Width of input data
0: 32-bit
1: 16-bit
ARG SIZE selects the number of bits used to represent input data.
If 32-bit data is selected, the CORDIC_WDATA register expects arguments in q1.31 format.
If 16-bit data is selected, the CORDIC_WDATA register expects arguments in q1.15 format.
The primary argument (ARG1) is written to the least significant half-word, and the secondary argument (ARG2) to the most significant half-word.

Bit 21 RESSIZE: Width of output data
0: 32-bit
1: 16-bit
RESSIZE selects the number of bits used to represent output data.
If 32-bit data is selected, the CORDIC_RDATA register contains results in q1.31 format.
If 16-bit data is selected, the least significant half-word of CORDIC_RDATA contains the primary result (RES1) in q1.15 format, and the most significant half-word contains the secondary result (RES2), also in q1.15 format.
Bit 20 **NARGS**: Number of arguments expected by the CORDIC_WDATA register
- 0: Only one 32-bit write (or two 16-bit values if ARGSIZE = 1) is needed for the next calculation.
- 1: Two 32-bit values must be written to the CORDIC_WDATA register to trigger the next calculation.
  Reads return the current state of the bit.

Bit 19 **NRES**: Number of results in the CORDIC_RDATA register
- 0: Only one 32-bit value (or two 16-bit values if RESSIZE = 1) is transferred to the CORDIC_RDATA register on completion of the next calculation. One read from CORDIC_RDATA resets the RRDY flag.
- 1: Two 32-bit values are transferred to the CORDIC_RDATA register on completion of the next calculation. Two reads from CORDIC_RDATA are necessary to reset the RRDY flag.
  Reads return the current state of the bit.

Bit 18 **DMAWEN**: Enable DMA write channel
- 0: Disabled. No DMA write requests are generated.
- 1: Enabled. Requests are generated on the DMA write channel whenever no operation is pending.
  This bit is set and cleared by software. A read returns the current state of the bit.

Bit 17 **DMAREN**: Enable DMA read channel
- 0: Disabled. No DMA read requests are generated.
- 1: Enabled. Requests are generated on the DMA read channel whenever the RRDY flag is set.
  This bit is set and cleared by software. A read returns the current state of the bit.

Bit 16 **IEN**: Enable interrupt
- 0: Disabled. No interrupt requests are generated.
- 1: Enabled. An interrupt request is generated whenever the RRDY flag is set.
  This bit is set and cleared by software. A read returns the current state of the bit.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **SCALE[2:0]**: Scaling factor
The value of this field indicates the scaling factor applied to the arguments and/or results. A value \(n\) implies that the arguments have been multiplied by a factor \(2^{-n}\), and/or the results need to be multiplied by \(2^n\). Refer to Section 20.3.2 for the applicability of the scaling factor for each function and the appropriate range.

Bits 7:4 **PRECISION[3:0]**: Precision required (number of iterations)
- 0: reserved
- 1 to 15: \((\text{Number of iterations})/4\)
  To determine the number of iterations needed for a given accuracy refer to Table 158.
  Note that for most functions, the recommended range for this field is 3 to 6.
20.4.2 CORDIC argument register (CORDIC_WDATA)

Address offset: 0x04
Reset value: 0xXXXX XXXX

Bits 31:0 ARG[31:0]: Function input arguments

This register is programmed with the input arguments for the function selected in the CORDIC_CSR register FUNC field.

If 32-bit format is selected (CORDIC_CSR.ARGSIZE = 0) and two input arguments are required (CORDIC_CSR.NARGS = 1), two successive writes are required to this register. The first writes the primary argument (ARG1), the second writes the secondary argument (ARG2).

If 32-bit format is selected and only one input argument is required (NARGS = 0), only one write is required to this register, containing the primary argument (ARG1).

If 16-bit format is selected (CORDIC_CSR.ARGSIZE = 1), one write to this register contains both arguments. The primary argument (ARG1) is in the lower half, ARG[15:0], and the secondary argument (ARG2) is in the upper half, ARG[31:16]. In this case, NARGS must be set to 0.

Refer to Section 20.3.2 for the arguments required by each function, and their permitted range.

When the required number of arguments has been written, the CORDIC evaluates the function designated by CORDIC_CSR.FUNC using the supplied input arguments, provided any previous calculation has completed. If a calculation is ongoing, the ARG1 and ARG 2 values are held pending until the calculation is completed and the results read. During this time, a write to the register cancels the pending operation and overwrite the argument data.
20.4.3 CORDIC result register (CORDIC_RDATA)

Address offset: 0x08
Reset value: 0x0000 0000

Bits 31:0 RES[31:0]: Function result
If 32-bit format is selected (CORDIC_CSR.RESSIZE = 0) and two output values are expected (CORDIC_CSR.NRES = 1), this register must be read twice when the RRDY flag is set. The first read fetches the primary result (RES1). The second read fetches the secondary result (RES2) and resets RRDY.
If 32-bit format is selected and only one output value is expected (NRES = 0), only one read of this register is required to fetch the primary result (RES1) and reset the RRDY flag.
If 16-bit format is selected (CORDIC_CSR.RESSIZE = 1), this register contains the primary result (RES1) in the lower half, RES[15:0], and the secondary result (RES2) in the upper half, RES[31:16]. In this case, NRES must be set to 0, and only one read performed.
A read from this register resets the RRDY flag in the CORDIC_CSR register.

20.4.4 CORDIC register map

Table 159. CORDIC register map and reset value

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | CORDIC_CSR    | RRDY | ARGSIZE | RESIZE | NARGS | NRES | DMAEN | DMAEN | IEN | SCALE [2:0] | PRECISION [3:0] | FUNC [3:0] |  |
|        | Reset value   | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x04   | CORDIC_WDATA  | ARG[31:0] |  |
|        | Reset value   | x x x x x x x x x x x x x x x x x x x x x x x x x |  |
| 0x08   | CORDIC_RDATA  | RES[31:0] |  |
|        | Reset value   | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Refer to Section 2.3 for the register boundary addresses.
21 Filter math accelerator (FMAC)

21.1 FMAC introduction

The filter math accelerator unit performs arithmetic operations on vectors. It comprises a multiplier/accumulator (MAC) unit, together with address generation logic which allows it to index vector elements held in local memory.

The unit includes support for circular buffers on input and output, which allows digital filters to be implemented. Both finite and infinite impulse response filters can be realized.

The unit allows frequent or lengthy filtering operations to be offloaded from the CPU, freeing up the processor for other tasks. In many cases it can accelerate such calculations compared to a software implementation, resulting in a speed-up of time critical tasks.

21.2 FMAC main features

- 16 x 16-bit multiplier
- 24 + 2-bit accumulator with addition and subtraction
- 16-bit input and output data
- 256 x 16-bit local memory
- Up to three areas can be defined in memory for data buffers (two input, one output), defined by programmable base address pointers and associated size registers
- Input and output buffers can be circular
- Filter functions: FIR, IIR (direct form 1)
- Vector functions: Dot product, convolution, correlation
- AHB slave interface
- DMA read and write data channels
21.3  FMAC functional description

21.3.1  General description

The FMAC is shown in Figure 95.

The unit is built around a fixed point multiplier and accumulator (MAC). The MAC can take two 16-bit input signed values from memory, multiply them together and add them to the contents of the accumulator. The address of the input values in memory is determined using a set of pointers. These pointers can be loaded, incremented, decremented or reset by the internal hardware. The pointer and MAC operations are controlled by a built-in sequencer in order to execute the requested operation.

To calculate a dot product, the two input vectors are loaded into the local memory by the processor or DMA controller, and the requested operation is selected and started. Each pair of input vector elements is fetched from memory, multiplied together and accumulated. When all the vector elements have been processed, the contents of the accumulator are stored in the local memory, from where they can be read out by the processor or DMA.

The finite impulse response (FIR) filter operation (also known as convolution) consists in repeatedly calculating the dot product of the coefficient vector and a vector of input samples, the latter being shifted by one sample delay, with the least recent sample being discarded and a new sample added, at each repetition.

The infinite impulse response (IIR) filter operation is the convolution of the feedback coefficients with the previous output samples, added to the result of the FIR convolution.

A more detailed description of the filter operations is given in Section 21.3.6: Filter functions.
21.3.2 Local memory and buffers

The unit contains a 256 x 16-bit read/write memory which is used for local storage:

- Input values (the elements of the input vectors) are stored in two buffers, X1 and X2.
- Output values (the results of the operations) are stored in another buffer, Y.
- The locations and sizes of the buffers are designated as follows:
  - x1_base: the base address of the X1 buffer
  - x2_base: the base address of the X2 buffer
  - y_base: the base address of the Y buffer
  - x1_buf_size: the number of 16-bit addresses allocated to the X1 buffer
  - x2_buf_size: the number of 16-bit addresses allocated to the X2 buffer
  - y_buf_size: the number of 16-bit addresses allocated to the Y buffer.

These parameters are programmed in the corresponding registers when configuring the unit.

The CPU (or DMA controller) can initialize the contents of each buffer using the Initialization functions (Section 21.3.5: Initialization functions) and writing to the write data register. The data is transferred to the location within the target buffer indicated by a write pointer. After each new write, the write pointer is incremented. When the write pointer reaches the end of the allocated buffer space, it wraps back to the base address. This feature is used to load the elements of a vector prior to an operation, or to initialize a filter and load filter coefficients.

Buffer configuration

The buffer sizes and base address offsets must be configured in the X1, X2 and Y buffer configuration registers. For each function, the required buffer size is specified in the function description in Section 21.3.6: Filter functions. The base addresses can be chosen anywhere in internal memory, provided that all buffers fit within the internal memory address range (0x00 to 0xFF), that is, base address + buffer size must be less than 256.

There is no constraint on the size and location of the buffers (they can overlap or even coincide exactly). For filter functions it is recommended not to overlap buffers as this can lead to erroneous behavior.

When circular buffer operation is required, an optional “headroom”, d, can be added to the buffer size. Furthermore, a watermark level can be set, to regulate the CPU or DMA activity. The value of d and the watermark level must be chosen according to the application performance requirements. For maximum throughput, the input buffer must never go empty, so d must be somewhat greater than the watermark level, allowing for any interrupt or DMA latency. On the other hand, if the input data can not be provided as fast as the unit can process them, the buffer can be allowed to empty waiting for the next data to be written, so d can be equal to the watermark level (to ensure that no overflow occurs on the input).

21.3.3 Input buffers

The X1 and X2 buffers are used to store data for input to the MAC. Each multiplication takes a value from the X1 buffer and a value from the X2 buffer and multiplies them together. A pointer in the control unit generates the read address offset (relative to the buffer base address) for each value. The pointers are managed by hardware according to the current function.
The X1 buffer can be used as a circular buffer, in which case new data are continually transferred into the input buffer whenever space is available. Pre-loading this buffer is optional for digital filters, since if no input samples have been written in the buffer when the operation is started, it is flagged as empty, which triggers the CPU or DMA to load new samples until there are enough to begin operation. Pre-loading is nevertheless useful in the case of a vector operation, that is, the input data is already available in system memory and circular operation is not required.
The X2 buffer can only be used in vector mode (that is not circular), and needs to be pre-loaded, except if the contents of the buffer do not change from one operation to the next. For filter functions, the X2 buffer is used to store the filter coefficients.

When operating as a circular buffer, the space allocated to the buffer (x1_buf_size) must generally be bigger than the number of elements in use for the current calculation, so that there are always new values available in the buffer. Figure 97 illustrates the layout of the buffer for a filter operation. While calculating an output sample y[n], the unit uses a set of N+1 input samples, x[n-N] to x[n]. When this is finished, the unit starts the calculation of y[n+1], using the set of input samples x[n-N+1] to x[n+1]. The least-recent input sample, x[n-N], drops out of the input set, and a new sample, x[n+1], is added to it.

The processor, or DMA controller, must ensure that the new sample x[n+1] is available in the buffer space when required. If not, the buffer is flagged as empty, which stalls the execution of the unit until a new sample is added. No underflow condition is signaled on the X1 buffer.
Note: If the flow of samples is controlled by a timer or other peripheral such as an ADC, the buffer regularly goes empty, since the filter processes each new sample faster than the source can provide it. This is an essential feature of filter operation.

If the number of free spaces in the buffer is less than the watermark threshold programmed in the FULL_WM bitfield of the FMAC_X1BUFCFG register, the buffer is flagged as full. As long as the full flag is not set, interrupts are generated, if enabled, to request more data for the buffer. The watermark allows several data to be transferred under one interrupt, without danger of overflow. Nevertheless, if an overflow does occur, the OVFL error flag is set and the write data is ignored. The write pointer is not incremented in the event of an overflow.

The operation of the X1 buffer during a filtering operation is illustrated in Figure 98. This example shows an 8-tap FIR filter with a watermark set to four.

Figure 98. Circular input buffer operation

21.3.4 Output buffer

The Y (output) buffer is used to store the output of an accumulation. Each new output value is stored in the buffer until it is read by the processor or DMA controller. Each time a read access is made to the read data register, the read data is fetched from the address indicated by the read pointer. This pointer is incremented after each read, and wraps back to the base address when it reaches the end of the allocated Y buffer space.
The Y buffer can also operate as a circular buffer. If the address for the next output value is the same as that indicated by the read pointer (an unread sample), then the buffer is flagged as full and execution stalled until the sample is read.

In the case of IIR filters, the Y buffer is used to store the set of M previous output samples, y[n-M] to y[n-1], used for calculating the next output sample y[n]. Each time a new sample is added to the set, the least recent sample y[n-M] drops out.

If the number of unread data in the buffer is less than the watermark threshold programmed in the EMPTY_WM bitfield of the FMAC_YBUFCFG register, the buffer is flagged as empty. As long as the empty flag is not set, interrupts or DMA requests are generated, if enabled, to request reads from the buffer. The watermark allows several data to be transferred under one interrupt, without danger of underflow. Nevertheless, if an underflow does occur, the UNFL error flag is set. In this case, the read pointer is not incremented and the read operation returns the content of the memory at the read pointer address.

The operation of the Y buffer in circular mode is illustrated in Figure 100. This example shows a 7-tap IIR filter with a watermark set to four.
21.3.5 Initialization functions

The following functions initialize the FMAC unit. They are triggered by writing the appropriate value in the FUNC bitfield of the FMAC_PARAM register, with the START bit set. The P and Q bitfields must also contain the appropriate parameter values for each function as detailed below. The R bitfield is not used. When the function completes, the START bit is automatically reset by hardware.

During initialization, it is recommended that the DMA requests and interrupts be disabled. The transfer of data into the FMAC memory can be done by software or by memory-to-memory DMA transfers, since no flow control is required.

Load X1 buffer

This function pre-loads the X1 buffer with N values, starting from the address in X1_BASE. Successive writes to the FMAC_WDATA register load the write data into the X1 buffer and increment the write address. The write pointer points to the address X1_BASE + N when the function completes.

The function can be used to pre-load the buffer with the elements of a vector, or to initialize the input storage elements of a filter.

Parameters

- The parameter P contains the number of values, N, to be loaded into the X1 buffer.
- The parameters Q and R are not used.

The function completes when N writes have been performed to the FMAC_WDATA register.
Load X2 buffer

This function pre-loads the X2 buffer with N + M values, starting from the address in X2_BASE. Successive writes to the FMAC_WDATA register load the write data into the X2 buffer and increment the write address.

The function can be used to pre-load the buffer with the elements of a vector, or the coefficients of a filter. In the case of an IIR, the N feed-forward and M feed-back coefficients are concatenated and loaded together into the X2 buffer. The total number of coefficients is equal to N + M. For an FIR, there are no feedback coefficients, so M = 0.

Parameters
- The parameter P contains the number of values, N, to be loaded into the X2 buffer starting from address X2_BASE.
- The parameter Q contains the number of values, M, to be loaded into the X2 buffer starting from address X2_BASE + N.
- The parameter R is not used.

The function completes when N + M writes have been performed to the FMAC_WDATA register.

Load Y buffer

This function pre-loads the Y buffer with N values, starting from the address in Y_BASE. Successive writes to the FMAC_WDATA register load the write data into the Y buffer and increment the write address. The read pointer points to the address Y_BASE + N when the function completes.

The function can be used to pre-load the feedback storage elements of an IIR filter.

Parameters
- The parameter P contains the number of values to be loaded into the Y buffer.
- The parameters Q and R are not used.

The function completes when N writes have been performed to the FMAC_WDATA register.

21.3.6 Filter functions

The following filter functions are supported by the FMAC unit. These functions are triggered by writing the corresponding value in the FUNC bitfield of the FMAC_PARAM register with the START bit set. The P, Q and R bitfields must also contain the appropriate parameter values for each function as detailed below. The filter functions continue to run until the START bit is reset by software.

Convolution (FIR filter)

\[ Y = B^T X \]

\[ y_n = 2^R \sum_{k=0}^{N} b_n x_{n-k} \]

This function performs a convolution of a vector B of length N+1 and a vector X of indefinite length. The elements of Y for incrementing values of n are calculated as the dot product,
\[ y_n = B \cdot X_n \], where \( X_n = [x_{n-N}, ..., x_n] \) is composed of the \( N+1 \) elements of \( X \) at indexes \( n - N \) to \( n \).

This function corresponds to a finite impulse response (FIR) filter, where vector \( B \) contains the filter coefficients and vector \( X \) the sampled data.

The structure of the filter (direct form) is shown in Figure 101.

**Figure 101. FIR filter structure**

Note that the cross correlation vector can be calculated by reversing the order of the coefficient vector \( B \).

**Input:**
- \( X1 \) buffer contains the elements of vector \( X \). It is a circular buffer of length \( N + 1 + d \).
- \( X2 \) buffer contains the elements of vector \( B \). It is a fixed buffer of length \( N + 1 \).

**Output:**
- \( Y \) buffer contains the output values, \( y_n \). It is a circular buffer of length \( d \).
Parameters:

- The parameter P contains the length, N+1, of the coefficient vector $B$ in the range [2:127].
- The parameter R contains the gain to be applied to the accumulator output. The value output to the $Y$ buffer is multiplied by $2^R$, where R is in the range [0:7]
- The parameter Q is not used.

The function completes when the START bit in the FMAC_PARAM register is reset by software.

IIR filter

$$Y = B^*X + A^*Y'$$

$$y_n = 2^R \cdot \left( \sum_{k=0}^{N} b_k x_{n-k} + \sum_{k=1}^{M} a_k y_{n-k} \right)$$

This function implements an infinite impulse response (IIR) filter. The filter output vector $Y$ is the convolution of a coefficient vector $B$ of length N+1 and a vector $X$ of indefinite length, plus the convolution of the delayed output vector $Y'$ with a second coefficient vector $A$, of length M. The elements of $Y$ for incrementing values of n are calculated as $y_n = B^*X_n + A^*Y_{n-1}$, where $X_n = [x_{n-N},...,x_n]$ comprises the N+1 elements of $X$ at indexes n - N to n, while $Y_{n-1} = [y_{n-M},...,y_{n-1}]$ comprises the M elements of $Y$ at indexes n - M to n - 1. The structure of the filter (direct form 1) is shown in Figure 102.
Figure 102. IIR filter structure (direct form 1)

Input:
- $X_1$ buffer contains the elements of vector $X$. It is a circular buffer of length $N + 1 + d$.
- $X_2$ buffer contains the elements of coefficient vectors $B$ and $A$ concatenated ($b_0, b_1, b_2, ..., b_N, a_1, a_2, ..., a_M$). It is a fixed buffer of length $M + N + 1$.

Output:
- $Y$ buffer contains the output values, $y_n$. It is a circular buffer of length $M + d$.

Parameters
- The parameter $P$ contains the length, $N + 1$, of the coefficient vector $B$ in the range [2:64].
- The parameter $Q$ contains the length, $M$, of the coefficient vector $A$ in the range [1:63].
- The parameter $R$ contains the gain to be applied to the accumulator output. The value output to the $Y$ buffer is multiplied by $2^R$, where $R$ is in the range [0:7].
The function completes when the START bit in the FMAC_PARAM register is reset by software.

21.3.7 Fixed point representation

The FMAC operates in fixed point signed integer format. Input and output values are q1.15.

In q1.15 format, numbers are represented by one sign bit and 15 fractional bits (binary decimal places). The numeric range is therefore -1 (0x8000) to 1 - 2^-15 (0x7FFF).

The accumulator has 26 bits, of which 22 are fractional and 4 are integer/sign (q4.22). This allows it to support partial accumulation sums in the range -8 (0x2000000) to +7.99999976 (0x1FFFFFF). A programmable gain from 0dB to 42dB in steps of 6dB can be applied at the output of the accumulator.

Note that the content of the accumulator is not saturated if the numeric range is exceeded. Partial sums whose value is greater than +7.99999976 or less than -8, wrap but this is harmless provided subsequent accumulations undo the wrapping. Nevertheless, the SAT flag in the FMAC_SR register is set if wrapping occurs, and generates an interrupt if the SATIEN bit is set in the FMAC_CR register. This helps in debugging the filter.

The data output by the accumulator can optionally be saturated, after application of the programmable gain, by setting the CLIPEN bit in the FMAC_CR register. If this bit is set, then any value which exceeds the numeric range of the q1.15 output, is set to 1 - 2^-15 or -1, according to the sign. If clipping is not enabled, the unused accumulator bits after applying the gain is simply truncated.

21.3.8 Implementing FIR filters with the FMAC

The FMAC supports FIR filters of length N, where N is the number of taps or coefficients. The minimum local memory requirement for a FIR filter of length N is 2N + 1:

- N coefficients
- N input samples
- 1 output sample

Since the local memory size is 256, the maximum value for N is 127.

If maximum throughput is required, it may be necessary to allocate a small amount of extra space, d1 and d2, to the input and output sample buffers respectively, to ensure that the filter never stalls waiting for a new input sample, or waiting for the output sample to be read. In this case, the local memory requirement is 2N + d1 + d2.

The buffers must be configured as follows:

- X1_BUF_SIZE = N + d1;
- X2_BUF_SIZE = N;
- Y_BUF_SIZE = d2 (or 1 if no extra space is required)

The buffer base addresses can be allocated anywhere, but the X2 buffer must not overlap with the others, or else the coefficients are overwritten. An example configuration is:

- X2_BASE = 0;
- X1_BASE = N;
- Y_BASE = 2N + d1
However, if the memory space is limited, the X1 and Y buffer areas can be overlapped, such that each output sample takes the place of the oldest input sample, which is no longer required:

- **X2_BASE = 0;**
- **X1_BASE = N;**
- **Y_BASE = N**

In this case, **Y_BUF_SIZE = X1_BUF_SIZE = N + d1**, so that the buffers remain in sync.

**Note:** The **FULL_WM** bitfield of X1 buffer configuration register must be programmed with a value less than or equal to \(\log_2(d1)\), otherwise the buffer is flagged full before \(N\) input samples have been written, and no more samples are requested. Similarly, the **EMPTY_WM** bitfield of the Y buffer configuration register must be less than or equal to \(\log_2(d2)\).

The filter coefficients **must** be pre-loaded into the X2 buffer, using the Load X2 Buffer function. The X1 buffer can optionally be pre-loaded with any number of samples up to a maximum of \(N\). There is no point in pre-loading the Y buffer, since for the FIR filter there is no feedback path.

After configuring and initializing the buffers, the FMAC_CR register must be programmed according to the method used for writing and reading data to and from the FMAC memory.

Three methods are supported:

- **Polling:** No DMA request or Interrupt request is generated. Software must check that the X1_FULL flag is low before writing to WDATA, or that the Y_EMPTY flag is low before reading from RDATA.
- **Interrupt:** The interrupt request is asserted while the X1_FULL flag is low, for writes, or when the Y_EMPTY flag is low, for reads.
- **DMA:** DMA requests are asserted on the DMA write channel while the X1_FULL flag is low, and on the read channel while the Y_EMPTY flag is low.

Different methods can be used for read and for write. However it is not recommended to use both interrupts and DMA requests for the same operation\(^{\text{a}}\). The valid combinations are listed in **Table 160**.

### Table 160. Valid combinations for read and write methods

<table>
<thead>
<tr>
<th>WIEN</th>
<th>RIEN</th>
<th>DMAWEN</th>
<th>DMAREN</th>
<th>Write</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Polling</td>
<td>Polling</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Polling</td>
<td>Interrupt</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Interrupt</td>
<td>Polling</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Interrupt</td>
<td>Interrupt</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Polling</td>
<td>DMA</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>DMA</td>
<td>Polling</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>DMA</td>
<td>DMA</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>DMA</td>
<td>Interrupt</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Interrupt</td>
<td>DMA</td>
</tr>
</tbody>
</table>

\(^{\text{a.}}\) If both interrupts and DMA requests are enabled then only DMA must perform the transfer.
The filter is started by writing to the FMAC_PARAM register with the following bitfield values:
- FUNC = 8 (FIR filter);
- P = N (number of coefficients);
- Q = “Don’t care”;
- R = Gain;
- START = 1;

If less than \( N + d - 2^{\text{FULL_WM}} \) values have been pre-loaded in the X1 buffer, the X1FULL flag remains low. If the WIEN bit is set in the FMAC_CR register, then the interrupt request is asserted immediately to request the processor to write \( 2^{\text{FULL_WM}} \) additional samples into the buffer, via the FMAC_WDATA register. It remains asserted until the X1FULL flag goes high in the FMAC_SR register. The interrupt service routine must check the X1FULL flag after every \( 2^{\text{FULL_WM}} \) writes to the FMAC_WDATA register, and repeat the transfer until the flag goes high. Similarly, if the DMAWEN bit is set in the FMAC_CR register, DMA write channel requests are generated until the X1FULL flag goes high.

The filter calculates the first output sample when at least N samples have been written into the X1 buffer (including any pre-loaded samples).

When \( 2^{\text{EMPTY_WM}} \) output samples have been written into the Y buffer, the YEMPTY flag in the FMAC_SR register goes low. If the RIEN bit is set in the FMAC_CR register, the interrupt request is asserted to request the processor to read \( 2^{\text{EMPTY_WM}} \) samples from the buffer, via the FMAC_RDATA register. It remains asserted until the YEMPTY flag goes high. The interrupt service routine must check the YEMPTY flag after every \( 2^{\text{EMPTY_WM}} \) reads from the FMAC_RDATA register, and repeat the transfer until the flag goes high. If the DMAREN bit is set in the FMAC_CR, DMA read channel requests are generated until the YEMPTY flag goes high.

The filter continues to operate in this fashion until it is stopped by the software resetting the START bit.

### Implementing IIR filters with the FMAC

The FMAC supports IIR filters of length N, where N is the number of feed-forward taps or coefficients. The number of feedback coefficients, M, can be any value from 1 to N-1. Only direct form 1 implementations can be realized, so filters designed for other forms need to be converted.

The minimum memory requirement for an IIR filter with N feed-forward coefficients and M feed-back coefficients is \( 2N + 2M \):
- \( N + M \) coefficients
- N input samples
- M output samples

If \( M = N-1 \), then the maximum filter length that can be implemented is \( N = 64 \).

As for the FIR, for maximum throughput, a small amount of additional space, \( d1 \) and \( d2 \), is allowed in the input and output buffer size respectively, making the total memory requirement \( 2M + 2N + d1 + d2 \).

The buffers must be configured as follows:
- \( X1\_BUF\_SIZE = N + d1 \);
- \( X2\_BUF\_SIZE = N + M \);
- \( Y\_BUF\_SIZE = M + d2 \);
The buffer base addresses can be allocated anywhere, but must not overlap. An example configuration is given below:

- X2_BASE = 0;
- X1_BASE = N + M;
- Y_BASE = 2N + M + d1;

**Note:** The FULL_WM bitfield of X1 buffer configuration register must be programmed with a value less than or equal to \(\log_2(d1)\), otherwise the buffer is flagged full before N input samples have been written, and no more samples are requested. Similarly, the EMPTY_WM bitfield of the Y buffer configuration register must be less than or equal to \(\log_2(d2)\).

The filter coefficients (N feed-forward followed by M feedback) must be pre-loaded into the X2 buffer, using the Load X2 Buffer function. The X1 buffer can optionally be pre-loaded with any number of samples up to a maximum of N. The Y buffer can optionally be pre-loaded with any number of values up to a maximum of M. This has the effect of initializing the feedback delay line.

After configuring the buffers, the FMAC_CR register must be programmed in the same way as for the FIR filter (see **Section 21.3.8: Implementing FIR filters with the FMAC**).

The filter is started by writing to the FMAC PARAM register with the following bitfield values:

- FUNC = 9 (IIR filter);
- P = N (number of feed-forward coefficients);
- Q = M (number of feed-back coefficients);
- R = Gain;
- START = 1;

If less than N + d - 2^{FULL_WM} values have been pre-loaded in the X1 buffer, the X1FULL flag remains low. If the WIEN bit is set in the FMAC_CR register, then the interrupt request is asserted immediately to request the processor to write 2^{FULL_WM} additional samples into the buffer, via the FMAC_WDATA register. It remains asserted until the X1FULL flag goes high in the FMAC_SR register. The interrupt service routine must check the X1FULL flag after every 2^{FULL_WM} writes to the FMAC_WDATA register, and repeat the transfer until the flag goes high. Similarly, if the DMAWEN bit is set in the FMAC_CR register, DMA write channel requests are generated until the X1FULL flag goes high.

The filter calculates the first output sample when at least N samples have been written into the X1 buffer (including any pre-loaded samples). The first sample is calculated using the first N samples in the X1 buffer, and the first M samples in the Y buffer (whether or not they are preloaded. The first output sample is written into the Y buffer at Y_BASE + M.

When 2^{EMPTY_WM} new output samples have been written into the Y buffer, the YEMPTY flag in the FMAC_SR register goes low. If the RIEN bit is set in the FMAC_CR register, the interrupt request is asserted to request the processor to read 2^{EMPTY_WM} samples from the buffer, via the FMAC_RDATA register. It remains asserted until the YEMPTY flag goes high. The interrupt service routine must check the YEMPTY flag after every 2^{EMPTY_WM} reads from the FMAC_RDATA register, and repeat the transfer until the flag goes high. If the DMAREN bit is set in the FMAC_CR, DMA read channel requests are generated until the YEMPTY flag goes high.

The filter continues to operate in this fashion until it is stopped by the software resetting the START bit.
21.3.10 Examples of filter initialization

Figure 103. X1 buffer initialization

The example in Figure 103 illustrates an X1 buffer pre-load with four samples (P = 4). The buffer size is six (X1_BUF_SIZE = 6). The initialization is launched by programming the FMAC_PARAM register with the START bit set. The four samples are then written to FMAC_WDATA, and transferred into local memory from X1_BASE onwards. The START bit resets after the fourth sample has been written. At this point, the X1 buffer contains the four samples, in order of writing, and the write pointer (next empty space) is at X1_BASE + 0x4.
21.3.11 Examples of filter operation

The example in Figure 104 illustrates the beginning of a filter operation. The filter has four taps (P=4). The X1 buffer size is six and the Y buffer size is two. The FULL_WM and EMPTY_WM bitfields are both set to 0. Prior to starting the filter, the X1 buffer has been pre-loaded with four samples, x[0:3] as in Figure 103. So the filter starts calculating the first output sample, y[0], immediately after the START bit is set. Since the X1FULL flag is not set (due to two uninitialized spaces in the X1 buffer), the interrupt is asserted straight away, to request new data. The processor writes two new samples, x[4] and x[5], to the FMAC_WDATA register, which are transferred to the empty locations in the X1 buffer.

In the mean time, the FMAC finishes calculating the first output sample, y[0], and writes it into the Y buffer, causing the Y_EMPTY flag to go low. At the same time, the x[0] sample is discarded, as it is no longer required, freeing up its location in memory (at X1_BASE). The FMAC can immediately start work on the second output sample, y[1], since all the required input samples x[1:5] are present in the X1 buffer.

Since the Y_EMPTY flag is low, the interrupt remains active after the processor finishes writing x[5]. The processor reads y[0] from the FMAC_RDATA register, freeing up its location in the Y buffer. There are now no samples in the output buffer since y[1] is still being calculated, so the Y_EMPTY flag goes high. Nevertheless, the interrupt remains active, because there is still free space in the X1 buffer, which the processor next fills with x[6], and so on.

Note: In this example, the processor can fill the input buffer more quickly than the FMAC can process them, so the X1_full flag regularly goes active. However, it struggles to read the Y buffer fast enough, so the FMAC stalls regularly waiting for space to be freed up in the Y buffer. This means the filter is not executing at maximum throughput. The reason is that the
filter length is small and the processor relatively slow, in this example. So increasing the Y buffer size would not help.

Figure 105. Filtering example 2

The example in Figure 105 illustrates the beginning of the same filter operation, but this time the filter has six taps (P=6). The X1 buffer size is six and the Y buffer size is two. The FULL_WM and EMPTY_WM bitfields are both set to 0. Prior to starting the filter, the X1 buffer has been pre-loaded with four samples, x[0:3] as in Figure 103. Because there are not enough samples in the input buffer, the X1FULL flag is not set, so the interrupt is asserted straight away, to request new data. The FMAC is stalled.

The processor writes two new samples, x[4] and x[5], to the FMAC_WDATA register, which are transferred to the empty locations in the X1 buffer. As soon as there are six unused samples in the X1 buffer, the X1_FULL flag goes active (since the buffer size is six), causing the interrupt to go inactive. The FMAC starts calculating the first output sample, y[0]. Since this requires all six input samples, there are no free spaces in the X1 buffer and so the X1_FULL flag remains active. Only when the FMAC finishes calculating y[0] and writes it into the Y buffer, can x[0] be discarded, freeing up a space in the X1 buffer, and deasserting X1_FULL. At the same time, the Y_EMPTY flag goes inactive. Both these flag states cause the interrupt to be asserted, requesting the processor to write a new input sample, first of all, and then read the output sample just calculated. The FMAC remains stalled until a new input sample is written.

In this example, the processor has to wait for the FMAC to finish calculating the current output sample, before it can write a new input sample, and therefore the X1 buffer regularly goes empty, stalling the FMAC. This can be avoided by allowing some extra space in the input buffer.
21.3.12 Filter design tips

The FMAC architecture imposes some constraints detailed below, on the design of digital filters.

1. Implementation of direct form 2, or transposed forms, is not efficient. Filters which have been designed for such forms must be converted to direct form 1.

2. Cascaded filters must either be combined into a single stage, or implemented as separate filters. In the latter case, multiple sets of filter coefficients can be pre-loaded into the memory, one set per stage, and only the X2_BASE address changed to select which set is used. The most efficient method of implementing a multi-stage filter is to pre-load a large X1 buffer with input samples, run the IIR filter function on it using the first stage coefficients, and store the output samples back in memory. Then change the X2_BASE pointer to point to the 2nd stage coefficients, and reload the input buffer with the output of the first stage (with a gain if required), before running the IIR function again. The procedure is repeated for all stages. Once the final stage samples have been transferred back into system memory, the input buffer can be loaded with the next set of input samples, and a new round of calculations started. Note that the N sample input buffer of each stage must be pre-loaded first of all with the N-1 last inputs from the previous round, plus one new sample, in order to keep continuity between each round. Similarly, the output buffer of each stage must be loaded with the last M samples from the previous round, for the same reason.

3. The use of direct form 1 for IIR designs can lead to large positive or negative partial sums in the accumulator, if for example a large step occurs on the input, or some of the filter coefficients’ absolute values are >1. Since the accumulator is limited to 26 bits, the biggest value that it can handle without wrapping (changing sign) is 0x1FFFFFF positive or 0x2000000 negative. This corresponds to 3.99999988 and -4 respectively in q3.23 fixed point format. Wrapping does not represent a problem provided the wrapping is “undone” before the end of the accumulation. However this is not always the case when a filter is starting up and can lead to unexpected results. Consider pre-loading the output buffer with suitable values to avoid this.

4. The IIR filter has feed-forward (numerator) coefficients \([b_0, b_1, ..., b_{N-1}]\), and feed-back (denominator) coefficients \([1, a_1, ..., a_M]\). Many IIR filters require some of the denominator coefficients to have an absolute value greater than 1 to achieve a steep roll-off in the frequency response. Given that the coefficients are coded in fixed point q1.15 format, this is not possible. Nevertheless, by scaling the denominator coefficients by a factor \(2^R\), such that \(2^{-R}[1, a_1, ..., a_M]\) are all less than 1, such filters can be implemented. However an inverse gain of \(2^R\) must be applied at the output of the accumulator to compensate the scaling. This has an adverse effect on the signal-to-noise ratio.
21.4 FMAC registers

21.4.1 FMAC X1 buffer configuration register (FMAC_X1BUFCFG)

Address offset: 0x00
Reset value: 0x0000 0000
Access: word access

This register can only be modified if START = 0 in the FMAC_PARAM register.

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL_WM[1:0]</td>
<td>25:24</td>
<td>rw</td>
<td>Watermark for buffer full flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Defines the threshold for setting the X1 buffer full flag when operating in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>circular mode. The flag is set if the number of free spaces in the buffer is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>less than $2^{\text{FULL_WM}}$.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Threshold = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Threshold = 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: Threshold = 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3: Threshold = 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Setting a threshold greater than 1 allows several data to be transferred into</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the buffer under one interrupt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Threshold must be set to 1 if DMA write requests are enabled (DMAWEN = 1 in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FMAC_CR register).</td>
</tr>
<tr>
<td>X1_BUF_SIZE[7:0]</td>
<td>23:16</td>
<td>rw</td>
<td>Allocated size of X1 buffer in 16-bit words</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The minimum buffer size is the number of feed-forward taps in the filter (+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the watermark threshold - 1).</td>
</tr>
<tr>
<td>X1_BASE[7:0]</td>
<td>15:8</td>
<td>rw</td>
<td>Base address of X1 buffer</td>
</tr>
</tbody>
</table>

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

Bits 25:24 FULL_WM[1:0]: Watermark for buffer full flag

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:8 X1_BUF_SIZE[7:0]: Allocated size of X1 buffer in 16-bit words

Bits 7:0 X1_BASE[7:0]: Base address of X1 buffer

21.4.2 FMAC X2 buffer configuration register (FMAC_X2BUFCFG)

Address offset: 0x04
Reset value: 0x0000 0000
Access: word access

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2_BUF_SIZE[7:0]</td>
<td>31:24</td>
<td>rw</td>
<td>Allocated size of X2 buffer in 16-bit words</td>
</tr>
<tr>
<td>X2_BASE[7:0]</td>
<td>15:8</td>
<td>rw</td>
<td>Base address of X2 buffer</td>
</tr>
</tbody>
</table>
21.4.3 **FMAC Y buffer configuration register (FMAC_YBUFCFG)**

Address offset: 0x08

Reset value: 0x0000 0000

Access: word access

This register can only be modified if START = 0 in the FMAC_PARAM register.

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

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:8 **X2_BUF_SIZE[7:0]**: Size of X2 buffer in 16-bit words

This bitfield can not be modified when a function is ongoing (START = 1).

Bits 7:0 **X2_BASE[7:0]**: Base address of X2 buffer

The X2 buffer base address can be modified while START=1, for example to change coefficient values. The filter must be stalled when doing this, since changing the coefficients while a calculation is ongoing affects the result.

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

Bits 25:24 **EMPTY_WM[1:0]**: Watermark for buffer empty flag

Defines the threshold for setting the Y buffer empty flag when operating in circular mode. The flag is set if the number of unread values in the buffer is less than 2\(^\text{EMPTY_WM}\).

0: Threshold = 1
1: Threshold = 2
2: Threshold = 4
3: Threshold = 8

Setting a threshold greater than 1 allows several data to be transferred from the buffer under one interrupt.

Threshold must be set to 1 if DMA read requests are enabled (DMAREN = 1 in FMAC_CR register).

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:8 **Y_BUF_SIZE[7:0]**: Size of Y buffer in 16-bit words

For FIR filters, the minimum buffer size is 1 (+ the watermark threshold). For IIR filters the minimum buffer size is the number of feedback taps (+ the watermark threshold).

Bits 7:0 **Y_BASE[7:0]**: Base address of Y buffer
21.4.4 FMAC parameter register (FMAC_PARAM)

Address offset: 0x0C
Reset value: 0x0000 0000
Access: word access

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

Bit 31 **START**: Enable execution
- 0: Stop execution
- 1: Start execution
Setting this bit triggers the execution of the function selected in the FUNC bitfield. Resetting it by software stops any ongoing function. For initialization functions, this bit is reset by hardware.

Bits 30:24 **FUNC[6:0]**: Function
- 0: Reserved
- 1: Load X1 buffer
- 2: Load X2 buffer
- 3: Load Y buffer
- 4 to 7: Reserved
- 8: Convolution (FIR filter)
- 9: IIR filter (direct form 1)
- 10 to 127: Reserved
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 23:16 **R[7:0]**: Input parameter R.
The value of this parameter is dependent on the function.
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 15:8 **Q[7:0]**: Input parameter Q.
The value of this parameter is dependent on the function.
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 7:0 **P[7:0]**: Input parameter P.
The value of this parameter is dependent on the function.
This bitfield can not be modified when a function is ongoing (START = 1)
21.4.5  FMAC control register (FMAC_CR)

Address offset: 0x10
Reset value: 0x0000 0000
Access: word access

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

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

Bit 16  **RESET**: Reset FMAC unit
This resets the write and read pointers, the internal control logic, the FMAC_SR register and the FMAC_PARAM register, including the START bit if active. Other register settings are not affected. This bit is reset by hardware.
0: Reset inactive
1: Reset active

Bit 15  **CLIPEN**: Enable clipping
0: Clipping disabled. Values at the output of the accumulator which exceed the q1.15 range, wrap.
1: Clipping enabled. Values at the output of the accumulator which exceed the q1.15 range are saturated to the maximum positive or negative value (+1 or -1) according to the sign.

Bits 14:10  Reserved, must be kept at reset value.

Bit 9  **DMAWEN**: Enable DMA write channel requests
0: Disable. No DMA requests are generated
1: Enable. DMA requests are generated while the X1 buffer is not full.
This bit can only be modified when START= 0 in the FMAC_PARAM register. A read returns the current state of the bit.

Bit 8  **DMAREN**: Enable DMA read channel requests
0: Disable. No DMA requests are generated
1: Enable. DMA requests are generated while the Y buffer is not empty.
This bit can only be modified when START= 0 in the FMAC_PARAM register. A read returns the current state of the bit.

Bits 7:5  Reserved, must be kept at reset value.

Bit 4  **SATIEN**: Enable saturation error interrupts
0: Disabled. No interrupts are generated upon saturation detection.
1: Enabled. An interrupt request is generated if the SAT flag is set
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 3  **UNFLIEN**: Enable underflow error interrupts
0: Disabled. No interrupts are generated upon underflow detection.
1: Enabled. An interrupt request is generated if the UNFL flag is set
This bit is set and cleared by software. A read returns the current state of the bit.
Bit 2 **OVFLIEN**: Enable overflow error interrupts  
0: Disabled. No interrupts are generated upon overflow detection.  
1: Enabled. An interrupt request is generated if the OVFL flag is set.  
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 1 **WIEN**: Enable write interrupt  
0: Disabled. No write interrupt requests are generated.  
1: Enabled. An interrupt request is generated while the X1 buffer FULL flag is not set.  
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 0 **RIEN**: Enable read interrupt  
0: Disabled. No read interrupt requests are generated.  
1: Enabled. An interrupt request is generated while the Y buffer EMPTY flag is not set.  
This bit is set and cleared by software. A read returns the current state of the bit.

### 21.4.6 FMAC status register (FMAC_SR)

Address offset: 0x14  
Reset value: 0x0000 0001  
Access: word access

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

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **SAT**: Saturation error flag  
Saturation occurs when the result of an accumulation exceeds the numeric range of the accumulator.  
0: No saturation detected  
1: Saturation detected. If the SATIEN bit is set, an interrupt is generated.  
This flag is cleared by a reset of the unit.

Bit 9 **UNFL**: Underflow error flag  
An underflow occurs when a read is made from FMAC_RDATA when no valid data is available in the Y buffer.  
0: No underflow detected  
1: Underflow detected. If the UNFLIEN bit is set, an interrupt is generated.  
This flag is cleared by a reset of the unit.

Bit 8 **OVFL**: Overflow error flag  
An overflow occurs when a write is made to FMAC_WDATA when no free space is available in the X1 buffer.  
0: No overflow detected  
1: Overflow detected. If the OVFLIEN bit is set, an interrupt is generated.  
This flag is cleared by a reset of the unit.

Bits 7:2 Reserved, must be kept at reset value.
Bit 1  X1FULL: X1 buffer full flag
The buffer is flagged as full if the number of available spaces is less than the FULL_WM
threshold. The number of available spaces is the difference between the write pointer and
the least recent sample currently in use.
0: X1 buffer not full. If the WIEN bit is set, the interrupt request is asserted until the flag is
set. If DMAWEN is set, DMA write channel requests are generated until the flag is set.
1: X1 buffer full.
This flag is set and cleared by hardware, or by a reset.
Note: *after the last available space in the X1 buffer is filled there is a delay of 3 clock cycles
before the X1FULL flag goes high. To avoid any risk of overflow it is recommended to
insert a software delay after writing to the X1 buffer before reading the FMAC_SR.
Alternatively, a FULL_WM threshold of 2 can be used.*

Bit 0  YEMPTY: Y buffer empty flag
The buffer is flagged as empty if the number of unread data is less than the EMPTY_WM
threshold. The number of unread data is the difference between the read pointer and the
current output destination address.
0: Y buffer not empty. If the RIEN bit is set, the interrupt request is asserted until the flag is
set. If DMAREN is set, DMA read channel requests are generated until the flag is set.
1: Y buffer empty.
This flag is set and cleared by hardware, or by a reset.
Note: *after the last sample is read from the Y buffer there is a delay of 3 clock cycles before
the YEMPTY flag goes high. To avoid any risk of underflow it is recommended to
insert a software delay after reading from the Y buffer before reading the FMAC_SR.
Alternatively, an EMPTY_WM threshold of 2 can be used.*

### 21.4.7  FMAC write data register (FMAC_WDATA)

<table>
<thead>
<tr>
<th>Address offset: 0x18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value: 0x0000 0000</td>
</tr>
<tr>
<td>Access: word and half-word access</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WDATA[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  WDATA[15:0]: Write data
When a write access to this register occurs, the write data are transferred to the address
offset indicated by the write pointer. The pointer address is automatically incremented after
each write access.
21.4.8 FMAC read data register (FMAC_RDATA)

Address offset: 0x1C
Reset value: 0x0000 0000
Access: word and half-word access

Bits 31:16 Reserved, must be kept at reset value.
Bits 15:0 \( RDATA[15:0] \): Read data

When a read access to this register occurs, the read data are the contents of the Y output buffer at the address offset indicated by the READ pointer. The pointer address is automatically incremented after each read access.

21.4.9 FMAC register map

Table 161 FMAC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>START</th>
<th>FUNC[6:0]</th>
<th>R[7:0]</th>
<th>Q[7:0]</th>
<th>P[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>FMAC_X1BUFCFG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Start value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>FMAC_X2BUFCFG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Start value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>FMAC_YBUFCFG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Start value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>FMAC_PARAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Start value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>FMAC_CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Start value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>FMAC_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Start value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>FMAC_WDATA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Start value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>FMAC_RDATA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Start value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

WDATA[15:0] for FMAC_WDATA

RDATA[15:0] for FMAC_RDATA
Refer to Section 2.3 for the register boundary addresses.
22 Flexible static memory controller (FSMC)

22.1 Introduction

The flexible static memory controller (FSMC) includes three memory controllers:
- The NOR/PSRAM memory controller
- The NAND memory controller
- The Synchronous DRAM (SDRAM/Mobile LPSDR SDRAM) controller

This memory controller is also named flexible memory controller (FMC).

22.2 FMC main features

The FMC functional block makes the interface with: synchronous and asynchronous static memories, SDRAM memories, and NAND flash memory. Its main purposes are:
- to translate AHB transactions into the appropriate external device protocol
- to meet the access time requirements of the external memory devices

All external memories share the addresses, data and control signals with the controller. Each external device is accessed by means of a unique chip select. The FMC performs only one access at a time to an external device.

The main features of the FMC controller are the following:
- Interface with static-memory mapped devices including:
  - Static random access memory (SRAM)
  - NOR flash memory/OneNAND flash memory
  - PSRAM (4 memory banks)
  - Ferroelectric RAM (FRAM)
  - NAND flash memory with ECC hardware to check up to 8 Kbytes of data
- Interface with synchronous DRAM (SDRAM/Mobile LPSDR SDRAM) memories
- Interface with parallel LCD modules, supporting Intel 8080 and Motorola 6800 modes.
- Burst mode support for faster access to synchronous devices such as NOR flash memory, PSRAM and SDRAM)
- Programmable continuous clock output for asynchronous and synchronous accesses
- 8-, 16-bit wide data bus
- Independent chip select control for each memory bank
- Independent configuration for each memory bank
- Write enable and byte lane select outputs for use with PSRAM, SRAM and SDRAM devices
- External asynchronous wait control
- Write FIFO with 16 x32-bit depth
- Cacheable Read FIFO with 6 x32-bit depth (6 x14-bit address tag) for SDRAM controller.
The Write FIFO is common to all memory controllers and consists of:

- A Write Data FIFO which stores the AHB data to be written to the memory (up to 32 bits) plus one bit for the AHB transfer (burst or not sequential mode)
- A Write Address FIFO which stores the AHB address (up to 28 bits) plus the AHB data size (up to 2 bits). When operating in burst mode, only the start address is stored except when crossing a page boundary (for PSRAM and SDRAM). In this case, the AHB burst is broken into two FIFO entries.

At startup the FMC pins must be configured by the user application. The FMC I/O pins which are not used by the application can be used for other purposes.

The FMC registers that define the external device type and associated characteristics are usually set at boot time and do not change until the next reset or power-up.

However, only a few bits can be changed on-the-fly:

- MBKEN, FMCEN, WEN bits in FMC_BCRx register
- ECCEN and PBKEN bits in the FMC_PCR register
- IFS, IRS and ILS bits in the FMC_SR register

Follow the below sequence to modify parameters while the FMC is enabled:

1. First disable the FMC controller to prevent further accesses to any memory controller while the register is modified.
2. Update all required configurations.
3. Enable the FMC controller again.

### 22.3 FMC block diagram

The FMC consists of the following main blocks:

- The AHB interface (including the FMC configuration registers)
- The NOR flash/PSRAM/SRAM controller
- The SDRAM controller

The block diagram is shown in the figure below.
22.4 AHB interface

The AHB slave interface allows internal CPUs and other bus master peripherals to access the external memories.

AHB transactions are translated into the external device protocol. In particular, if the selected external memory is 16- or 8-bit wide, 32-bit wide transactions on the AHB are split into consecutive 16- or 8-bit accesses. The FMC chip select (FMC_NEx) does not toggle between the consecutive accesses except in case of Access mode D when the Extended mode is enabled.

The FMC generates an AHB error in the following conditions:

- When reading or writing to a FMC bank (Bank 1 to 4) which is not enabled.
- When reading or writing to the NOR flash bank while the FACCEN bit is reset in the FMC_BCRx register.
- When writing to a write protected SDRAM bank (WP bit set in the SDRAM_SDCRx register).
- When the SDRAM address range is violated (access to reserved address range)
The effect of an AHB error depends on the AHB master which has attempted the R/W access:

- If the access has been attempted by the Cortex-M33 CPU, a hard fault interrupt is generated.
- If the access has been performed by a DMA controller, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

The AHB clock (HCLK) is the reference clock for the FMC.

### 22.4.1 Supported memories and transactions

#### General transaction rules

The requested AHB transaction data size can be 8-, 16- or 32-bit wide whereas the accessed external device has a fixed data width. This may lead to inconsistent transfers.

Therefore, some simple transaction rules must be followed:

- **AHB transaction size and memory data size are equal**
  
  There is no issue in this case.

- **AHB transaction size is greater than the memory size**
  
  In this case, the FMC splits the AHB transaction into smaller consecutive memory accesses to meet the external data width. The FMC chip select (FMC_NEx) does not toggle between the consecutive accesses. If the bus turnaround timings is configured to any other value than 0, the FMC chip select (FMC_NEx) toggles between the consecutive accesses. This feature is required when interfacing with FRAM memory.

- **AHB transaction size is smaller than the memory size**

  The transfer may or not be consistent depending on the type of external device:

  - **Accesses to devices that have the byte select feature (SRAM, ROM, PSRAM, SDRAM)**
    
    In this case, the FMC allows read/write transactions and accesses to the right data through its byte lanes NBL[1:0].
    
    Bytes to be written are addressed by NBL[1:0].
    
    All memory bytes are read (NBL[1:0] are driven low during read transaction) and the useless ones are discarded.

  - **Accesses to devices that do not have the byte select feature (NOR and NAND flash memories)**
    
    This situation occurs when a byte access is requested to a 16-bit wide flash memory. Since the device cannot be accessed in Byte mode (only 16-bit words can be read/written from/to the flash memory), Write transactions and Read transactions are allowed (the controller reads the entire 16-bit memory word and uses only the required byte).

#### Wrap support for NOR flash/PSRAM and SDRAM

Wrap burst mode for synchronous memories is not supported. The memories must be configured in Linear burst mode of undefined length.

#### Configuration registers

The FMC can be configured through a set of registers. Refer to Section 22.6.6, for a detailed description of the NOR flash/PSRAM controller registers. Refer to Section 22.7.7,
for a detailed description of the NAND flash registers and to Section 22.8.5 for a detailed description of the SDRAM controller registers.

22.5 External device address mapping

From the FMC point of view, the external memory is divided into fixed-size banks of 256 Mbytes each (see Figure 107):

- Bank 1 used to address up to 4 NOR flash memory or PSRAM devices. This bank is split into 4 NOR/PSRAM subbanks with 4 dedicated chip selects, as follows:
  - Bank 1 - NOR/PSRAM 1
  - Bank 1 - NOR/PSRAM 2
  - Bank 1 - NOR/PSRAM 3
  - Bank 1 - NOR/PSRAM 4
- Bank 3 used to address NAND flash memory devices. The MPU memory attribute for this space must be reconfigured by software to Device.
- Bank 4 and 5 used to address SDRAM devices (1 device per bank).

For each bank the type of memory to be used can be configured by the user application through the Configuration register.

![Figure 107. FMC memory banks](MSv69582V1)
22.5.1 NOR/PSRAM address mapping

HADDR[27:26] bits are used to select one of the four memory banks as shown in Table 162.

<table>
<thead>
<tr>
<th>HADDR[27:26]</th>
<th>Selected bank</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Bank 1 - NOR/PSRAM 1</td>
</tr>
<tr>
<td>01</td>
<td>Bank 1 - NOR/PSRAM 2</td>
</tr>
<tr>
<td>10</td>
<td>Bank 1 - NOR/PSRAM 3</td>
</tr>
<tr>
<td>11</td>
<td>Bank 1 - NOR/PSRAM 4</td>
</tr>
</tbody>
</table>

1. HADDR are internal AHB address lines that are translated to external memory.

The HADDR[25:0] bits contain the external memory address. Since HADDR is a byte address whereas the memory is addressed at word level, the address actually issued to the memory varies according to the memory data width, as shown in the following table.

<table>
<thead>
<tr>
<th>Memory width</th>
<th>Data address issued to the memory</th>
<th>Maximum memory capacity (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit</td>
<td>HADDR[25:0]</td>
<td>64 Mbytes x 8 = 512 Mbits</td>
</tr>
<tr>
<td>16-bit</td>
<td>HADDR[25:1] &gt;&gt; 1</td>
<td>64 Mbytes/2 x 16 = 512 Mbits</td>
</tr>
</tbody>
</table>

1. In case of a 16-bit external memory width, the FMC internally uses HADDR[25:1] to generate the address for external memory FMC_A[24:0]. Whatever the external memory width, FMC_A[0] must be connected to external memory address A[0].

22.5.2 NAND flash memory address mapping

The NAND bank is divided into memory areas as indicated in Table 164.

<table>
<thead>
<tr>
<th>Start address</th>
<th>End address</th>
<th>FMC bank</th>
<th>Memory space</th>
<th>Timing register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x8800 0000</td>
<td>0x8BFF FFFF</td>
<td>Bank 3</td>
<td>Attribute</td>
<td>FMC_PATT (0x8C)</td>
</tr>
<tr>
<td>0x8000 0000</td>
<td>0x83FF FFFF</td>
<td>Bank 3</td>
<td>Common</td>
<td>FMC_PMEM (0x88)</td>
</tr>
</tbody>
</table>

For NAND flash memory, the common and attribute memory spaces are subdivided into three sections (see in Table 165 below) located in the lower 256 Kbytes:

- Data section (first 64 Kbytes in the common/attribute memory space)
- Command section (second 64 Kbytes in the common/attribute memory space)
- Address section (next 128 Kbytes in the common/attribute memory space)
The application software uses the 3 sections to access the NAND flash memory:

- **To sending a command to NAND flash memory**, the software must write the command value to any memory location in the command section.

- **To specify the NAND flash address that must be read or written**, the software must write the address value to any memory location in the address section. Since an address can be 4 or 5 bytes long (depending on the actual memory size), several consecutive write operations to the address section are required to specify the full address.

- **To read or write data**, the software reads or writes the data from/to any memory location in the data section.

Since the NAND flash memory automatically increments addresses, there is no need to increment the address of the data section to access consecutive memory locations.

### 22.5.3 SDRAM address mapping

The HADDR[28] bit (internal AHB address line 28) is used to select one of the two memory banks as indicated in Table 166.

<table>
<thead>
<tr>
<th>HADDR[28]</th>
<th>Selected bank</th>
<th>Control register</th>
<th>Timing register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SDRAM Bank1</td>
<td>FMC_SDCR1</td>
<td>FMC_SDTR1</td>
</tr>
<tr>
<td>1</td>
<td>SDRAM Bank2</td>
<td>FMC_SDCR2</td>
<td>FMC_SDTR2</td>
</tr>
</tbody>
</table>

The following table shows SDRAM mapping for a 13-bit row, a 11-bit column and a 4 internal bank configuration.

<table>
<thead>
<tr>
<th>Memory width</th>
<th>Internal bank</th>
<th>Row address</th>
<th>Column address</th>
<th>Maximum memory capacity (Mbytes)</th>
</tr>
</thead>
</table>

1. When interfacing with a 16-bit memory, the FMC internally uses the HADDR[11:1] internal AHB address lines to generate the external address. Whatever the memory width, FMC_A[0] has to be connected to the external memory address A[0].

2. The AutoPrecharge is not supported. FMC_A[10] must be connected to the external memory address A[10] but it will be always driven 'low'.
The HADDR[27:0] bits are translated to external SDRAM address depending on the SDRAM controller configuration:

- Data size: 8 or 16 bits
- Row size: 11, 12 or 13 bits
- Column size: 8, 9, 10 or 11 bits
- Number of internal banks: two or four internal banks

The following tables show the SDRAM address mapping versus the SDRAM controller configuration.

### Table 168. SDRAM address mapping with 8-bit data bus width\(^{(1)}\)(\(^{(2)}\))

<table>
<thead>
<tr>
<th>Row size configuration</th>
<th>HADDR(AHB Internal Address Lines)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>27 26 25 24 23 22 21</td>
</tr>
<tr>
<td>11-bit row size</td>
<td></td>
</tr>
<tr>
<td>12-bit row size</td>
<td></td>
</tr>
<tr>
<td>13-bit row size</td>
<td></td>
</tr>
</tbody>
</table>

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be set to ‘0’.
2. Access to Reserved (Res.) address range generates an AHB error.
The FMC generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM, FRAM and ROM
  - 8 bits
  - 16 bits
- PSRAM (CellularRAM™)
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed
- NOR flash memory
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed

The FMC outputs a unique chip select signal, NE[4:1], per bank. All the other signals (addresses, data and control) are shared.
The FMC supports a wide range of devices through a programmable timings among which:

- Programmable wait states (up to 15)
- Programmable bus turnaround cycles (up to 15)
- Programmable output enable and write enable delays (up to 15)
- Independent read and write timings and protocol to support the widest variety of memories and timings
- Programmable continuous clock (FMC_CLK) output.

The FMC Clock (FMC_CLK) is a submultiple of the HCLK clock. It can be delivered to the selected external device either during synchronous accesses only or during asynchronous and synchronous accesses depending on the CCKEN bit configuration in the FMC_BCR1 register:

- If the CCLKEN bit is reset, the FMC generates the clock (CLK) only during synchronous accesses (Read/write transactions).
- If the CCLKEN bit is set, the FMC generates a continuous clock during asynchronous and synchronous accesses. To generate the FMC_CLK continuous clock, Bank 1 must be configured in Synchronous mode (see Section 22.6.6: NOR/PSRAM controller registers). Since the same clock is used for all synchronous memories, when a continuous output clock is generated and synchronous accesses are performed, the AHB data size has to be the same as the memory data width (MWID) otherwise the FMC_CLK frequency is changed depending on AHB data transaction (refer to Section 22.6.5: Synchronous transactions for FMC_CLK divider ratio formula).

The size of each bank is fixed and equal to 64 Mbytes. Each bank is configured through dedicated registers (see Section 22.6.6: NOR/PSRAM controller registers).

The programmable memory parameters include access times (see Table 170) and support for wait management (for PSRAM and NOR flash accessed in Burst mode).

<table>
<thead>
<tr>
<th>Table 170. Programmable NOR/PSRAM access parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter</strong></td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Address setup</td>
</tr>
<tr>
<td>Address hold</td>
</tr>
<tr>
<td>NBL setup</td>
</tr>
<tr>
<td>Data setup</td>
</tr>
<tr>
<td>Data hold</td>
</tr>
<tr>
<td>Burst turn</td>
</tr>
</tbody>
</table>
Table 170. Programmable NOR/PSRAM access parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Access mode</th>
<th>Unit</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock divide ratio</td>
<td>Number of AHB clock cycles (HCLK) to build one memory clock cycle (CLK)</td>
<td>Synchronous</td>
<td>AHB clock cycle (HCLK)</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>Data latency</td>
<td>Number of clock cycles to issue to the memory before the first data of the burst</td>
<td>Synchronous</td>
<td>Memory clock cycle (CLK)</td>
<td>2</td>
<td>17</td>
</tr>
</tbody>
</table>

22.6.1 External memory interface signals

Table 171, Table 172 and Table 173 list the signals that are typically used to interface with NOR flash memory, SRAM and PSRAM.

Note: The prefix “N” identifies the signals that are active low.

NOR flash memory, non-multiplexed I/Os

Table 171. Non-multiplexed I/O NOR flash memory

<table>
<thead>
<tr>
<th>FMC signal name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>O</td>
<td>Clock (for synchronous access)</td>
</tr>
<tr>
<td>A[25:0]</td>
<td>O</td>
<td>Address bus</td>
</tr>
<tr>
<td>D[15:0]</td>
<td>I/O</td>
<td>Bidirectional data bus</td>
</tr>
<tr>
<td>NE[x]</td>
<td>O</td>
<td>Chip select, x = 1..4</td>
</tr>
<tr>
<td>NOE</td>
<td>O</td>
<td>Output enable</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NL(=NADV)</td>
<td>O</td>
<td>Latch enable (this signal is called address valid, NADV, by some NOR flash devices)</td>
</tr>
<tr>
<td>NWAIT</td>
<td>I</td>
<td>NOR flash wait input signal to the FMC</td>
</tr>
</tbody>
</table>

The maximum capacity is 512 Mbits (26 address lines).

NOR flash memory, 16-bit multiplexed I/Os

Table 172. 16-bit multiplexed I/O NOR flash memory

<table>
<thead>
<tr>
<th>FMC signal name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>O</td>
<td>Clock (for synchronous access)</td>
</tr>
<tr>
<td>AD[15:0]</td>
<td>I/O</td>
<td>16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)</td>
</tr>
<tr>
<td>NE[x]</td>
<td>O</td>
<td>Chip select, x = 1..4</td>
</tr>
<tr>
<td>NOE</td>
<td>O</td>
<td>Output enable</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
</tbody>
</table>
The maximum capacity is 512 Mbits.

### PSRAM/FRAM/SRAM, non-multiplexed I/Os

The maximum capacity is 512 Mbits.

### PSRAM, 16-bit multiplexed I/Os

The maximum capacity is 512 Mbits.

### Table 172. 16-bit multiplexed I/O NOR flash memory (continued)

<table>
<thead>
<tr>
<th>FMC signal name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL(=NADV)</td>
<td>O</td>
<td>Latch enable (this signal is called address valid, NADV, by some NOR flash devices)</td>
</tr>
<tr>
<td>NWAIT</td>
<td>I</td>
<td>NOR flash wait input signal to the FMC</td>
</tr>
</tbody>
</table>

### Table 173. Non-multiplexed I/Os PSRAM/SRAM

<table>
<thead>
<tr>
<th>FMC signal name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>O</td>
<td>Clock (only for PSRAM synchronous access)</td>
</tr>
<tr>
<td>A[25:0]</td>
<td>O</td>
<td>Address bus</td>
</tr>
<tr>
<td>D[15:0]</td>
<td>I/O</td>
<td>Data bidirectional bus</td>
</tr>
<tr>
<td>NE[x]</td>
<td>O</td>
<td>Chip select, x = 1..4 (called NCE by PSRAM (CellularRAM™ i.e. CRAM))</td>
</tr>
<tr>
<td>NOE</td>
<td>O</td>
<td>Output enable</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NL(= NADV)</td>
<td>O</td>
<td>Address valid only for PSRAM input (memory signal name: NADV)</td>
</tr>
<tr>
<td>NWAIT</td>
<td>I</td>
<td>PSRAM wait input signal to the FMC</td>
</tr>
<tr>
<td>NBL[1:0]</td>
<td>O</td>
<td>Byte lane output. Byte 0 and Byte 1 control (upper and lower byte enable)</td>
</tr>
</tbody>
</table>

### Table 174. 16-Bit multiplexed I/O PSRAM

<table>
<thead>
<tr>
<th>FMC signal name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>O</td>
<td>Clock (for synchronous access)</td>
</tr>
<tr>
<td>AD[15:0]</td>
<td>I/O</td>
<td>16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)</td>
</tr>
<tr>
<td>NE[x]</td>
<td>O</td>
<td>Chip select, x = 1..4 (called NCE by PSRAM (CellularRAM™ i.e. CRAM))</td>
</tr>
<tr>
<td>NOE</td>
<td>O</td>
<td>Output enable</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NL(= NADV)</td>
<td>O</td>
<td>Address valid PSRAM input (memory signal name: NADV)</td>
</tr>
<tr>
<td>NWAIT</td>
<td>I</td>
<td>PSRAM wait input signal to the FMC</td>
</tr>
<tr>
<td>NBL[1:0]</td>
<td>O</td>
<td>Byte lane output. Byte 0 and Byte 1 control (upper and lower byte enable)</td>
</tr>
</tbody>
</table>

The maximum capacity is 512 Mbits (26 address lines).
### 22.6.2 Supported memories and transactions

*Table 175* below shows an example of the supported devices, access modes and transactions when the memory data bus is 16-bit wide for NOR flash memory, PSRAM and SRAM. The transactions not allowed (or not supported) by the FMC are shown in gray in this example.

<table>
<thead>
<tr>
<th>Device</th>
<th>Mode</th>
<th>R/W</th>
<th>AHB data size</th>
<th>Memory data size</th>
<th>Allowed/not allowed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NOR flash</strong> (muxed I/Os and nonmuxed I/Os)</td>
<td>Asynchronous</td>
<td>R</td>
<td>8</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>8</td>
<td>16</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>-</td>
<td>16</td>
<td>N</td>
<td>Mode is not supported</td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>R</td>
<td>8</td>
<td>16</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>R</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>R</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td><strong>PSRAM</strong> (multiplexed I/Os and non-multiplexed I/Os)</td>
<td>Asynchronous</td>
<td>R</td>
<td>8</td>
<td>16</td>
<td>Y</td>
<td>Use of byte lanes NBL[1:0]</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>-</td>
<td>16</td>
<td>N</td>
<td>Mode is not supported</td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>R</td>
<td>8</td>
<td>16</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>R</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>R</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>W</td>
<td>8</td>
<td>16</td>
<td>Y</td>
<td>Use of byte lanes NBL[1:0]</td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>W</td>
<td>16/32</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td><strong>SRAM and ROM</strong></td>
<td>Asynchronous</td>
<td>R</td>
<td>8 / 16</td>
<td>16</td>
<td>Y</td>
<td>Use of byte lanes NBL[1:0]</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses Use of byte lanes NBL[1:0]</td>
</tr>
</tbody>
</table>
22.6.3 General timing rules

Signals synchronization

- All controller output signals change on the rising edge of the internal clock (HCLK)
- In Synchronous mode (read or write), all output signals change on the rising edge of HCLK. Whatever the CLKDIV value, all outputs change as follows:
  - NOEL/NWEL/ NEL/NADV/ NADVH /NBLL/ Address valid outputs change on the falling edge of FMC_CLK clock.
  - NOEH/ NWEH / NEH/ NOEH/NBLH/ Address invalid outputs change on the rising edge of FMC_CLK clock.

22.6.4 NOR flash/PSRAM controller asynchronous transactions

Asynchronous static memories (NOR flash, PSRAM, SRAM, FRAM)

- Signals are synchronized by the internal clock HCLK. This clock is not issued to the memory
- The FMC always samples the data before de-asserting the NOE signal. This guarantees that the memory data hold timing constraint is met (minimum Chip Enable high to data transition is usually 0 ns)
- If the Extended mode is enabled (EXTMOD bit is set in the FMC_BCRx register), up to four extended modes (A, B, C and D) are available. It is possible to mix A, B, C and D modes for read and write operations. For example, read operation can be performed in mode A and write in mode B.
- If the Extended mode is disabled (EXTMOD bit is reset in the FMC_BCRx register), the FMC can operate in mode 1 or mode 2 as follows:
  - Mode 1 is the default mode when SRAM/PSRAM memory type is selected (MTYP = 0x0 or 0x01 in the FMC_BCRx register)
  - Mode 2 is the default mode when NOR memory type is selected (MTYP = 0x10 in the FMC_BCRx register).
Mode 1 - SRAM/FRAM/PSRAM (CRAM)

The next figures show the read and write transactions for the supported modes followed by the required configuration of FMC_BCRx, and FMC_BTRx/FMC_BWTRx registers.

**Figure 108. Mode 1 read access waveforms**

**Figure 109. Mode 1 write access waveforms**
The DATAHLD time at the end of the read and write transactions guarantee the address and data hold time after the NOE/NWE rising edge. The DATAST value must be greater than zero (DATAST > 0).

### Table 176. FMC_BCRx bitfields (mode 1)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:24</td>
<td>Reserved</td>
<td>0x0000</td>
</tr>
<tr>
<td>23:22</td>
<td>NBLSET[1:0]</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>Don’t care</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>As needed, exclude 0x2 (NOR flash memory)</td>
</tr>
<tr>
<td>1</td>
<td>MUXE</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 177. FMC_BTRx bitfields (mode 1)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>DATAHLD</td>
<td>Duration of the data hold phase (DATAHLD HCLK cycles for read accesses, DATAHLD+1 HCLK cycles for write accesses).</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN HCLK).</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST HCLK cycles).</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>
Mode A - SRAM/FRAM/PSRAM (CRAM) OE toggling

Figure 110. Mode A read access waveforms

1. NBL[1:0] are driven low during the read access

Figure 111. Mode A write access waveforms

The differences compared with Mode 1 are the toggling of NOE and the independent read and write timings.
### Table 178. FMC_BCRx bitfields (mode A)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:24</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>23:22</td>
<td>NBLSET[1:0]</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCHWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x1</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>Don’t care</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>As needed, exclude 0x2 (NOR flash memory)</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 179. FMC_BTRx bitfields (mode A)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>DATAHLD</td>
<td>Duration of the data hold phase (DATAHLD HCLK cycles for read accesses).</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN HCLK).</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST HCLK cycles) for read accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>
Table 180. FMC_BWTRx bitfields (mode A)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>DATAHL</td>
<td>Duration of the data hold phase (DATAHL+1 HCLK cycles for write accesses).</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN HCLK).</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST HCLK cycles) for write accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>

Mode 2/B - NOR flash

Figure 112. Mode 2 and mode B read access waveforms
The differences with mode 1 are the toggling of NWE and the independent read and write timings when extended mode is set (mode B).
### Table 181. FMC_BCRx bitfields (mode 2/B)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:24</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>23:22</td>
<td>NBLSET[1:0]</td>
<td>Don’t care</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCEWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x1 for mode B, 0x0 for mode 2</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>0x2 (NOR flash memory)</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 182. FMC_BTRx bitfields (mode 2/B)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>DATAHLD</td>
<td>Duration of the data hold phase (DATAHLD HCLK cycles for read accesses and DATAHLD+1 HCLK cycles for write accesses when Extended mode is disabled).</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x1 if Extended mode is set</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN HCLK).</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the access second phase (DATAST HCLK cycles) for read accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the access first phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>
Table 183. FMC_BWTRx bitfields (mode 2/B)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>DATAHLD</td>
<td>Duration of the data hold phase (DATAHLD+1 HCLK cycles for write accesses).</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x1 if Extended mode is set</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN HCLK).</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the access second phase (DATAST HCLK cycles) for write accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the access first phase (ADDSET HCLK cycles) for write accesses.</td>
</tr>
</tbody>
</table>

Note: The FMC_BWTRx register is valid only if the Extended mode is set (mode B), otherwise its content is don’t care.

Mode C - NOR flash - OE toggling

Figure 115. Mode C read access waveforms
The differences compared with mode 1 are the toggling of NOE and the independent read and write timings.

Table 184. FMC_BCRx bitfields (mode C)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:24</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>23:22</td>
<td>NBLSET[1:0]</td>
<td>Don’t care</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCEWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x1</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>
### Table 184. FMC_BCRx bitfields (mode C) (continued)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>0x02 (NOR flash memory)</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 185. FMC_BTRx bitfields (mode C)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>DATAHLD</td>
<td>Duration of the data hold phase (DATAHLD HCLK cycles for read accesses).</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x2</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>0x0</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>0x0</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN HCLK).</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST HCLK cycles) for read accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>

### Table 186. FMC_BWTRx bitfields (mode C)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>DATAHLD</td>
<td>Duration of the data hold phase (DATAHLD+1 HCLK cycles for write accesses).</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x2</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN HCLK).</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST HCLK cycles) for write accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>
Mode D - asynchronous access with extended address

Figure 117. Mode D read access waveforms

- A[25:0]
- NADV
- NBL[x:0]
- NEx
- NOE
- NWE
- Data bus
- Data driven by memory

Memory transaction

- NBLSET HCLK cycles
- ADDSET HCLK cycles
- ADHLD HCLK cycles
- DATAST HCLK cycles
- DATAHLD HCLK cycles
The differences with mode 1 are the toggling of NOE that goes on toggling after NADV changes and the independent read and write timings.

Table 187. FMC_BCRx bitfields (mode D)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:24</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>23:22</td>
<td>NBLSET[1:0]</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x1</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
</tbody>
</table>
### Table 187. FMC_BCRx bitfields (mode D) (continued)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>Set according to memory support</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>As needed</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 188. FMC_BTRx bitfields (mode D)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>DATAHLD</td>
<td>Duration of the data hold phase (DATAHLD HCLK cycles for read accesses).</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x3</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN HCLK).</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST HCLK cycles) for read accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Duration of the middle phase of the read access (ADDHLD HCLK cycles)</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 1.</td>
</tr>
</tbody>
</table>

### Table 189. FMC_BWTRx bitfields (mode D)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>DATAHLD</td>
<td>Duration of the data hold phase (DATAHLD+1 HCLK cycles for write accesses).</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x3</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN HCLK).</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST HCLK cycles).</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Duration of the middle phase of the write access (ADDHLD HCLK cycles)</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 1.</td>
</tr>
</tbody>
</table>
Muxed mode - multiplexed asynchronous access to NOR flash memory

Figure 119. Muxed read access waveforms

- A[25:16]
- NADV
- NBL[x:0]
- NEx
- NOE
- NWE (High)
- AD[15:0]
- NBLSET, ADDSET, ADDHLD, DATAHLD, DATAST (HCLK cycles)
The difference with mode D is the drive of the lower address byte(s) on the data bus.

### Table 190. FMC_BCRx bitfields (Muxed mode)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:24</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>23:22</td>
<td>NBLSET[1:0]</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
</tbody>
</table>
WAIT management in asynchronous accesses

If the asynchronous memory asserts the WAIT signal to indicate that it is not yet ready to accept or to provide data, the ASYNCPWAIT bit has to be set in FMC_BCRx register.

If the WAIT signal is active (high or low depending on the WAITPOL bit), the second access phase (Data setup phase), programmed by the DATAST bits, is extended until WAIT becomes inactive. Unlike the data setup phase, the first access phases (Address setup and Address hold phases), programmed by the ADDSET and ADDHLD bits, are not WAIT sensitive and so they are not prolonged.
The data setup phase must be programmed so that WAIT can be detected 4 HCLK cycles before the end of the memory transaction. The following cases must be considered:

1. The memory asserts the WAIT signal aligned to NOE/NWE which toggles:
   \[ \text{DATAST} \geq (4 \times \text{HCLK}) + \text{max\_wait\_assertion\_time} \]

2. The memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):
   
   If
   \[ \text{max\_wait\_assertion\_time} > \text{address\_phase} + \text{hold\_phase} \]
   
   Then:
   \[ \text{DATAST} \geq (4 \times \text{HCLK}) + (\text{max\_wait\_assertion\_time} - \text{address\_phase} - \text{hold\_phase}) \]
   
   Otherwise
   \[ \text{DATAST} \geq 4 \times \text{HCLK} \]
   
   where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

*Figure 121* and *Figure 122* show the number of HCLK clock cycles that are added to the memory access phase after WAIT is released by the asynchronous memory (independently of the above cases).

*Figure 121. Asynchronous wait during a read access waveforms*

1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.
CellularRAM™ (PSRAM) refresh management

The CellularRAM™ does not enable maintaining the chip select signal (NE) low for longer than the $t_{CEM}$ timing specified for the memory device. This timing can be programmed in the FMC_PCSCNTR register. It defines the maximum duration of the NE low pulse in HCLK cycles for asynchronous accesses and FMC_CLK cycles for synchronous accesses.

### 22.6.5 Synchronous transactions

The memory clock, FMC_CLK, is a submultiple of HCLK. It depends on the value of CLKDIV and the MWID/AHB data size, following the formula given below:

Whatever MWID size: 16 or 8-bit, the FMC_CLK divider ratio is always defined by the programmed CLKDIV value.

Example:
- If CLKDIV=1, MWID = 16 bits, AHB data size=8 bits, FMC_CLK=HCLK/2.

NOR flash memories specify a minimum time from NADV assertion to CLK high. To meet this constraint, the FMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs in the middle of the NADV low pulse.

**Data latency versus NOR memory latency**

The data latency is the number of cycles to wait before sampling the data. The DATLAT value must be consistent with the latency value specified in the NOR flash configuration.
register. The FMC does not include the clock cycle when NADV is low in the data latency count.

**Caution:** Some NOR flash memories include the NADV Low cycle in the data latency count, so that the exact relation between the NOR flash latency and the FMC DATLAT parameter can be either:

- NOR flash latency = (DATLAT + 2) CLK clock cycles
- or NOR flash latency = (DATLAT + 3) CLK clock cycles

Some recent memories assert NWAIT during the latency phase. In such cases DATLAT can be set to its minimum value. As a result, the FMC samples the data and waits long enough to evaluate if the data are valid. Thus the FMC detects when the memory exits latency and real data are processed.

Other memories do not assert NWAIT during latency. In this case the latency must be set correctly for both the FMC and the memory, otherwise invalid data are mistaken for good data, or valid data are lost in the initial phase of the memory access.

**Single-burst transfer**

When the selected bank is configured in Burst mode for synchronous accesses, if for example an AHB single-burst transaction is requested on 16-bit memories, the FMC performs a burst transaction of length 1 (if the AHB transfer is 16 bits), or length 2 (if the AHB transfer is 32 bits) and de-assert the chip select signal when the last data is strobed.

Such transfers are not the most efficient in terms of cycles compared to asynchronous read operations. Nevertheless, a random asynchronous access would first require to re-program the memory access mode, which would altogether last longer.

**Cross boundary page for CellularRAM™ 1.5**

CellularRAM™ 1.5 does not allow burst access to cross the page boundary. The FMC controller is used to split automatically the burst access when the memory page size is reached by configuring the CPSIZE bits in the FMC_BCR1 register following the memory page size.

**Wait management**

For synchronous NOR flash memories, NWAIT is evaluated after the programmed latency period, which corresponds to (DATLAT+2) CLK clock cycles.

If NWAIT is active (low level when WAITPOL = 0, high level when WAITPOL = 1), wait states are inserted until NWAIT is inactive (high level when WAITPOL = 0, low level when WAITPOL = 1).

When NWAIT is inactive, the data is considered valid either immediately (bit WAITCFG = 1) or on the next clock edge (bit WAITCFG = 0).

During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the chip select and output enable signals valid. It does not consider the data as valid.

In Burst mode, there are two timing configurations for the NOR flash NWAIT signal:

- The flash memory asserts the NWAIT signal one data cycle before the wait state (default after reset).
- The flash memory asserts the NWAIT signal during the wait state
The FMC supports both NOR flash wait state configurations, for each chip select, thanks to the WAITCFG bit in the FMC_BCRx registers (x = 0..3).

**Figure 123. Wait configuration waveforms**

Memory transaction = burst of 4 half words
Figure 124. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)

Table 192. FMC_BCRx bitfields (Synchronous multiplexed read mode)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:24</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>23:22</td>
<td>NBLSET[1:0]</td>
<td>Don't care</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>No effect on synchronous read</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCEWAIT</td>
<td>0x0</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>To be set to 1 if the memory supports this feature, to be kept at 0 otherwise</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>No effect on synchronous read</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>To be set according to memory</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
</tbody>
</table>

1. Byte lane outputs (NBL are not shown; for NOR access, they are held high, and, for PSRAM (CRAM) access, they are held low.
### Table 192. FMC_BCRx bitfields (Synchronous multiplexed read mode) (continued)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>To be set according to memory</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x1</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>Set according to memory support (NOR flash memory)</td>
</tr>
<tr>
<td>5-4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3-2</td>
<td>MTYP</td>
<td>0x1 or 0x2</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>As needed</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 193. FMC_BTRx bitfields (Synchronous multiplexed read mode)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>DATAHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Data latency</td>
</tr>
</tbody>
</table>
| 23-20      | CLKDIV   | 0x0 to get CLK = HCLK  
|            |          | 0x1 to get CLK = 2 × HCLK  
|            |          | ..           |
| 19-16      | BUSTURN  | Time between NEx high to NEx low (BUSTURN HCLK). |
| 15-8       | DATAST   | Don’t care   |
| 7-4        | ADDHLD   | Don’t care   |
| 3-0        | ADDSET   | Don’t care   |
1. The memory must issue NWAIT signal one cycle in advance, accordingly WAITCFG must be programmed to 0.
2. Byte Lane (NBL) outputs are not shown, they are held low while NEx is active.

Table 194. FMC_BCRx bitfields (Synchronous multiplexed write mode)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:24</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>23:22</td>
<td>NBLSET[1:0]</td>
<td>Don’t care</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x1</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>As needed (0x1 for CRAM 1.5)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>0x0</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x0</td>
</tr>
</tbody>
</table>
### Table 194. FMC_BCRx bitfields (Synchronous multiplexed write mode) (continued)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>To be set to 1 if the memory supports this feature, to be kept at 0 otherwise.</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>0x1</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>0x0</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>to be set according to memory</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>no effect on synchronous write</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>Set according to memory support</td>
</tr>
<tr>
<td>5-4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3-2</td>
<td>MTYP</td>
<td>0x1</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>As needed</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 195. FMC_BTRx bitfields (Synchronous multiplexed write mode)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-30</td>
<td>DATAHLDD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>27-24</td>
<td>DATLAT</td>
<td>Data latency</td>
</tr>
</tbody>
</table>
| 23-20      | CLKDIV     | 0x0 to get CLK = HCLK  
               |            | 0x1 to get CLK = 2 × HCLK |
| 19-16      | BUSTURN    | Time between NEx high to NEx low (BUSTURN HCLK) |
| 15-8       | DATAST     | Don’t care   |
| 7-4        | ADDHLDD    | Don’t care   |
| 3-0        | ADDSET     | Don’t care   |
22.6.6  NOR/PSRAM controller registers

SRAM/NOR-flash chip-select control register for bank x (FMC_BCRx) (x = 1 to 4)

Address offset: 0x00 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 30DB, 0x0000 30D2, 0x0000 30D2, 0x0000 30D2

This register contains the control information of each memory bank, used for SRAMs, PSRAM, FRAM and NOR flash memories.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>FMCEN: FMC controller enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit enables or disables the FMC controller.</td>
</tr>
<tr>
<td>0: Disable the FMC controller</td>
<td></td>
</tr>
<tr>
<td>1: Enable the FMC controller</td>
<td></td>
</tr>
<tr>
<td>Note: The FMCEN bit of the FMC_BCR2..4 registers is don’t care. It is only enabled through the FMC_BCR1 register.</td>
<td></td>
</tr>
</tbody>
</table>

| Bit 30:24 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bit 23:22</th>
<th>NBLSET[1:0]: Byte lane (NBL) setup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>These bits configure the NBL setup timing from NBLx low to chip select NEx low.</td>
</tr>
<tr>
<td>00: NBL setup time is 0 AHB clock cycle</td>
<td></td>
</tr>
<tr>
<td>01: NBL setup time is 1 AHB clock cycle</td>
<td></td>
</tr>
<tr>
<td>10: NBL setup time is 2 AHB clock cycles</td>
<td></td>
</tr>
<tr>
<td>11: NBL setup time is 3 AHB clock cycles</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 21</th>
<th>WFDIS: Write FIFO disable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit disables the Write FIFO used by the FMC controller.</td>
</tr>
<tr>
<td>0: Write FIFO enabled (Default after reset)</td>
<td></td>
</tr>
<tr>
<td>1: Write FIFO disabled</td>
<td></td>
</tr>
<tr>
<td>Note: The WFDIS bit of the FMC_BCR2..4 registers is don’t care. It is only enabled through the FMC_BCR1 register.</td>
<td></td>
</tr>
</tbody>
</table>
Bit 20 **CCLKEN**: Continuous clock enable
This bit enables the FMC_CLK clock output to external memory devices.
0: The FMC_CLK is only generated during the synchronous memory access (read/write transaction). The FMC_CLK clock ratio is specified by the programmed CLKDIV value in the FMC_BCRx register (default after reset).
1: The FMC_CLK is generated continuously during asynchronous and synchronous access. The FMC_CLK clock is activated when the CCLKEN is set.

Note: The CCLKEN bit of the FMC_BCR2..4 registers is don't care. It is only enabled through the FMC_BCR1 register. Bank 1 must be configured in Synchronous mode to generate the FMC_CLK continuous clock.

Note: If CCLKEN bit is set, the FMC_CLK clock ratio is specified by CLKDIV value in the FMC_BTR1 register. CLKDIV in FMC_BWTR1 is don't care.

Note: If the Synchronous mode is used and CCLKEN bit is set, the synchronous memories connected to other banks than Bank 1 are clocked by the same clock (the CLKDIV value in the FMC_BTR2..4 and FMC_BWTR2..4 registers for other banks has no effect.)

Bit 19 **CBURSTRW**: Write burst enable
For PSRAM (CRAM) operating in Burst mode, the bit enables synchronous accesses during write operations. The enable bit for synchronous read accesses is the BURSTEN bit in the FMC_BCRx register.
0: Write operations are always performed in Asynchronous mode.
1: Write operations are performed in Synchronous mode.

Bits 18:16 **CPSIZE[2:0]**: CRAM page size
These are used for CellularRAM™ 1.5 which does not allow burst access to cross the address boundaries between pages. When these bits are configured, the FMC controller splits automatically the burst access when the memory page size is reached (refer to memory datasheet for page size).
000: No burst split when crossing page boundary (default after reset)
001: 128 bytes
010: 256 bytes
011: 512 bytes
100: 1024 bytes
Others: reserved

Bit 15 **ASYNCWAIT**: Wait signal during asynchronous transfers
This bit enables/disables the FMC to use the wait signal even during an asynchronous protocol.
0: NWAIT signal is not taken in to account when running an asynchronous protocol (default after reset).
1: NWAIT signal is taken in to account when running an asynchronous protocol.

Bit 14 **EXTMOD**: Extended mode enable
This bit enables the FMC to program the write timings for non multiplexed asynchronous accesses inside the FMC_BWTR register, thus resulting in different timings for read and write operations.
0: values inside FMC_BWTR register are not taken into account (default after reset)
1: values inside FMC_BWTR register are taken into account

Note: When the Extended mode is disabled, the FMC can operate in mode 1 or mode 2 as follows:
- Mode 1 is the default mode when the SRAM/PSRAM memory type is selected (MTYP = 0x0 or 0x01)
- Mode 2 is the default mode when the NOR memory type is selected (MTYP = 0x10).
Bit 13 **WAITEN**: Wait enable bit
This bit enables/disables wait-state insertion via the NWAIT signal when accessing the memory in Synchronous mode.
0: NWAIT signal is disabled (its level not taken into account, no wait state inserted after the programmed flash latency period).
1: NWAIT signal is enabled (its level is taken into account after the programmed latency period to insert wait states if asserted) (default after reset).

Bit 12 **WREN**: Write enable bit
This bit indicates whether write operations are enabled/disabled in the bank by the FMC.
0: Write operations are disabled in the bank by the FMC, an AHB error is reported.
1: Write operations are enabled for the bank by the FMC (default after reset).

Bit 11 **WAITCFG**: Wait timing configuration
The NWAIT signal indicates whether the data from the memory are valid or if a wait state must be inserted when accessing the memory in Synchronous mode. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait state or during the wait state:
0: NWAIT signal is active one data cycle before wait state (default after reset).
1: NWAIT signal is active during wait state (not used for PSRAM).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **WAITPOL**: Wait signal polarity bit
Defines the polarity of the wait signal from memory used for either in Synchronous or Asynchronous mode.
0: NWAIT active low (default after reset)
1: NWAIT active high

Bit 8 **BURSTEN**: Burst enable bit
This bit enables/disables synchronous accesses during read operations. It is valid only for synchronous memories operating in Burst mode.
0: Burst mode disabled (default after reset). Read accesses are performed in Asynchronous mode.
1: Burst mode enabled. Read accesses are performed in Synchronous mode.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FACCEN**: Flash access enable
Enables NOR flash memory access operations.
0: Corresponding NOR flash memory access is disabled.
1: Corresponding NOR flash memory access is enabled (default after reset).

Bits 5:4 **MWID[1:0]**: Memory data bus width
Defines the external memory device width, valid for all type of memories.
00: 8 bits
01: 16 bits (default after reset)
10: reserved
11: reserved

Bits 3:2 **MTYP[1:0]**: Memory type
Defines the type of external memory attached to the corresponding memory bank.
00: SRAM/FRAM (default after reset for Bank 2...4)
01: PSRAM (CRAM) / FRAM
10: NOR flash/OneNAND flash (default after reset for Bank 1)
11: reserved
Bit 1 **MUXEN**: Address/data multiplexing enable bit
- When this bit is set, the address and data values are multiplexed on the data bus, valid only with NOR and PSRAM memories:
  0: Address/data non multiplexed
  1: Address/data multiplexed on databus (default after reset)

Bit 0 **MBKEN**: Memory bank enable bit
- Enables the memory bank. After reset Bank1 is enabled, all others are disabled. Accessing a disabled bank causes an ERROR on AHB bus.
  0: Corresponding memory bank is disabled.
  1: Corresponding memory bank is enabled.

### SRAM/NOR-flash chip-select timing register for bank x (FMC_BTRx)

Address offset: 0x04 + 0x8 * (x - 1), (x = 1 to 4)
Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank, used for SRAMs, PSRAM and NOR flash memories. If the EXTMOD bit is set in the FMC_BCRx register, then this register is partitioned for write and read access, that is, 2 registers are available: one to configure read accesses (this register) and one to configure write accesses (FMC_BWTRx registers).

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

#### Bits 31:30 **DATAHLD[1:0]**: Data hold phase duration
- These bits are written by software to define the duration of the data hold phase in HCLK cycles (refer to Figure 108 to Figure 120), used in asynchronous accesses:
  For read accesses
  - 00: DATAHLD phase duration = 0 × HCLK clock cycle (default)
  - 01: DATAHLD phase duration = 1 × HCLK clock cycle
  - 10: DATAHLD phase duration = 2 × HCLK clock cycle
  - 11: DATAHLD phase duration = 3 × HCLK clock cycle
  For write accesses
  - 00: DATAHLD phase duration = 1 × HCLK clock cycle (default)
  - 01: DATAHLD phase duration = 2 × HCLK clock cycle
  - 10: DATAHLD phase duration = 3 × HCLK clock cycle
  - 11: DATAHLD phase duration = 4 × HCLK clock cycle

#### Bits 29:28 **ACCMOD[1:0]**: Access mode
- Specifies the asynchronous access modes as shown in the timing diagrams. These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.
  - 00: Access mode A
  - 01: Access mode B
  - 10: Access mode C
  - 11: Access mode D
Bits 27:24 **DATLAT[3:0]**: (see note below bit descriptions): Data latency for synchronous memory  
For synchronous access with read/write Burst mode enabled (BURSTEN / CBURSTRW bits set), defines the number of memory clock cycles (+2) to issue to the memory before reading/writing the first data:  
This timing parameter is not expressed in HCLK periods, but in FMC_CLK periods.  
For asynchronous access, this value is don't care.  
0000: Data latency of 2 CLK clock cycles for first burst access  
1111: Data latency of 17 CLK clock cycles for first burst access (default value after reset)

Bits 23:20 **CLKDIV[3:0]**: Clock divide ratio (for FMC_CLK signal)  
Defines the period of FMC_CLK clock output signal, expressed in number of HCLK cycles:  
0000: FMC_CLK period= 1x HCLK period  
0001: FMC_CLK period = 2 × HCLK periods  
0010: FMC_CLK period = 3 × HCLK periods  
1111: FMC_CLK period = 16 × HCLK periods (default value after reset)  
In asynchronous NOR flash, SRAM or PSRAM accesses, this value is don't care.  
**Note:** Refer to Section 22.6.5: Synchronous transactions for FMC_CLK divider ratio formula

Bits 19:16 **BUSTURN[3:0]**: Bus turnaround phase duration  
These bits are written by software to add a delay at the end of current read or write transaction to next transaction on the same bank.  
This delay is used to match the minimum time between consecutive transactions \(t_{EHEL}\) from NEx high to NEx low) and the maximum time needed by the memory to free the data bus after a read access \(t_{EHQZ}\), chip enable high to output Hi-Z). This delay is recommended for mode D and muxed mode. For non-muxed memory, the bus turnaround delay can be set to minimum value.  
\[(BUSTURN + 1)HCLK period \geq \max(t_{EHEL} \text{ min}, t_{EHQZ} \text{ max})\]  
For FRAM memories, the bus turnaround delay must be configured to match the minimum IPC (precharge time) timings. The bus turnaround delay is inserted between any consecutive transactions on the same bank (read/read, write/write, read/write and write/read) to match the IPC memory timing. The chip select is toggling between any consecutive accesses.  
\[(BUSTURN + 1)HCLK period \geq t_{PC} \text{ min}\]  
0000: BUSTURN phase duration = 1 HCLK clock cycle added  
...  
1111: BUSTURN phase duration = 16 x HCLK clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]**: Data-phase duration  
These bits are written by software to define the duration of the data phase (refer to Figure 108 to Figure 120), used in asynchronous accesses:  
0000 0000: Reserved  
0000 0001: DATAST phase duration = 1 × HCLK clock cycles  
0000 0010: DATAST phase duration = 2 × HCLK clock cycles  
...  
1111 1111: DATAST phase duration = 255 × HCLK clock cycles (default value after reset)  
For each memory type and access mode data-phase duration, refer to the respective figure (Figure 108 to Figure 120).  
Example: Mode 1, write access, DATAST=1: Data-phase duration= DATAST+1 = 2 HCLK clock cycles.  
**Note:** In synchronous accesses, this value is don't care.
Bits 7:4 ADDHLD[3:0]: Address-hold phase duration
These bits are written by software to define the duration of the address hold phase (refer to Figure 108 to Figure 120), used in mode D or multiplexed accesses:
0000: Reserved
0001: ADDHLD phase duration = 1 × HCLK clock cycle
0010: ADDHLD phase duration = 2 × HCLK clock cycle
... 1111: ADDHLD phase duration = 15 × HCLK clock cycles (default value after reset)
For each access mode address-hold phase duration, refer to the respective figure (Figure 108 to Figure 120).
Note: In synchronous accesses, this value is not used, the address hold phase is always 1 memory clock period duration.

Bits 3:0 ADDSET[3:0]: Address setup phase duration
These bits are written by software to define the duration of the address setup phase (refer to Figure 108 to Figure 120), used in SRAMs, ROMs, asynchronous NOR flash and PSRAM:
0000: ADDSET phase duration = 0 × HCLK clock cycle
... 1111: ADDSET phase duration = 15 × HCLK clock cycles (default value after reset)
For each access mode address setup phase duration, refer to the respective figure (Figure 108 to Figure 120).
Note: In synchronous accesses, this value is don’t care.
In Muxed mode or mode D, the minimum value for ADDSET is 1.
In mode 1 and PSRAM memory, the minimum value for ADDSET is 1.

Note: PSRams (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to prolong the latency as needed.
With PSRams (CRAMs) the filled DATLAT must be set to 0, so that the FMC exits its latency phase soon and starts sampling NWAIT from memory, then starts to read or write when the memory is ready.
This method can be used also with the latest generation of synchronous flash memories that issue the NWAIT signal, unlike older flash memories (check the datasheet of the specific flash memory being used).

SRAM/NOR-flash write timing registers x (FMC_BWTRx)
Address offset: 0x104 + 0x8 × (x - 1), (x = 1 to 4)
Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank. It is used for SRAMs, PSRams and NOR flash memories. When the EXTMOD bit is set in the FMC_BCRx register, then this register is active for write access.
Bits 31:30  **DATAHLD[1:0]**: Data hold phase duration
These bits are written by software to define the duration of the data hold phase in HCLK cycles (refer to Figure 108 to Figure 120), used in asynchronous write accesses:
- 00: DATAHLD phase duration = 1 × HCLK clock cycle (default)
- 01: DATAHLD phase duration = 2 × HCLK clock cycle
- 10: DATAHLD phase duration = 3 × HCLK clock cycle
- 11: DATAHLD phase duration = 4 × HCLK clock cycle

Bits 29:28  **ACCMOD[1:0]**: Access mode.
Specifies the asynchronous access modes as shown in the next timing diagrams. These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.
- 00: Access mode A
- 01: Access mode B
- 10: Access mode C
- 11: Access mode D

Bits 27:20 Reserved, must be kept at reset value.

Bits 19:16  **BUSTURN[3:0]**: Bus turnaround phase duration
These bits are written by software to add a delay at the end of current write transaction to next transaction on the same bank.
For FRAM memories, the bus turnaround delay must be configured to match the minimum tPC (precharge time) timings. The bus turnaround delay is inserted between any consecutive transactions on the same bank (read/read, write/write, read/write and write/read). The chip select is toggling between any consecutive accesses.

(BUSTURN + 1)HCLK period ≥ tPC min
- 0000: BUSTURN phase duration = 1 HCLK clock cycle added
- 1111: BUSTURN phase duration = 16 × HCLK clock cycles added (default value after reset)

Bits 15:8  **DATAST[7:0]**: Data-phase duration.
These bits are written by software to define the duration of the data phase (refer to Figure 108 to Figure 120), used in asynchronous SRAM, PSRAM and NOR flash memory accesses:
- 0000 0000: Reserved
- 0000 0001: DATAST phase duration = 1 × HCLK clock cycles
- 0000 0010: DATAST phase duration = 2 × HCLK clock cycles
- ...
- 1111 1111: DATAST phase duration = 255 × HCLK clock cycles (default value after reset)

Bits 7:4  **ADDHLD[3:0]**: Address-hold phase duration.
These bits are written by software to define the duration of the address hold phase (refer to Figure 117 to Figure 120), used in asynchronous multiplexed accesses:
- 0000: Reserved
- 0001: ADDHLD phase duration = 1 × HCLK clock cycle
- 0010: ADDHLD phase duration = 2 × HCLK clock cycle
- ...
- 1111: ADDHLD phase duration = 15 × HCLK clock cycles (default value after reset)

**Note:** In synchronous NOR flash accesses, this value is not used, the address hold phase is always 1 flash clock period duration.
Bits 3:0 **ADDSET[3:0]**: Address setup phase duration.
These bits are written by software to define the duration of the address setup phase in HCLK cycles (refer to Figure 108 to Figure 120), used in asynchronous accesses:
0000: ADDSET phase duration = 0 × HCLK clock cycle
...
1111: ADDSET phase duration = 15 × HCLK clock cycles (default value after reset)
*Note:* In synchronous accesses, this value is not used, the address setup phase is always 1 flash clock period duration. In muxed mode, the minimum ADDSET value is 1.

**PSRAM chip select counter register (FMC_PCSCNTR)**
Address offset: 0x20
Reset value: 0x0000 0000
This register contains the PSRAM chip select counter value for Synchronous and Asynchronous modes. The chip select counter is common to all banks and can be enabled separately on each bank. During PSRAM read or write accesses, this value is loaded into a timer which is decremented while the NE signal is held low. When the timer reaches 0, the PSRAM controller splits the current access, toggles NE to allow PSRAM device refresh, and restarts a new access. The programmed counter value guarantees a maximum NE pulse width (tCEM) as specified for PSRAM devices. The counter is reloaded and starts decrementing each time a new access is started by a transition of NE from high to low.

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

**CSCOUNT[15:0]**

Bits 31:20 Reserved, must be kept at reset value.

- **Bit 19** **CNTB4EN**: Counter Bank 4 enable
  This bit enables the chip select counter for PSRAM/NOR Bank 4.
  0: Counter disabled for Bank 4
  1: Counter enabled for Bank 4

- **Bit 18** **CNTB3EN**: Counter Bank 3 enable
  This bit enables the chip select counter for PSRAM/NOR Bank 3.
  0: Counter disabled for Bank 3.
  1: Counter enabled for Bank 3

- **Bit 17** **CNTB2EN**: Counter Bank 2 enable
  This bit enables the chip select counter for PSRAM/NOR Bank 2.
  0: Counter disabled for Bank 2
  1: Counter enabled for Bank 2

- **Bit 16** **CNTB1EN**: Counter Bank 1 enable
  This bit enables the chip select counter for PSRAM/NOR Bank 1.
  0: Counter disabled for Bank 1
  1: Counter enabled for Bank 1
The FMC generates the appropriate signal timings to drive the following types of device:

- 8- and 16-bit NAND flash memories

The NAND bank is configured through dedicated registers (Section 22.7.7). The programmable memory parameters include access timings (shown in Table 196) and ECC configuration.

### 22.7 NAND flash controller

The FMC generates the appropriate signal timings to drive the following types of device:

- 8- and 16-bit NAND flash memories

The NAND bank is configured through dedicated registers (Section 22.7.7). The programmable memory parameters include access timings (shown in Table 196) and ECC configuration.

**Table 196. Programmable NAND flash access parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Access mode</th>
<th>Unit</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory setup time</td>
<td>Number of clock cycles (HCLK) required to set up the address before the command assertion</td>
<td>Read/Write</td>
<td>AHB clock cycle (HCLK)</td>
<td>1</td>
<td>255</td>
</tr>
<tr>
<td>Memory wait</td>
<td>Minimum duration (in HCLK clock cycles) of the command assertion</td>
<td>Read/Write</td>
<td>AHB clock cycle (HCLK)</td>
<td>2</td>
<td>255</td>
</tr>
<tr>
<td>Memory hold</td>
<td>Number of clock cycles (HCLK) during which the address must be held (as well as the data if a write access is performed) after the command de-assertion</td>
<td>Read/Write</td>
<td>AHB clock cycle (HCLK)</td>
<td>1</td>
<td>254</td>
</tr>
<tr>
<td>Memory databus high-Z</td>
<td>Number of clock cycles (HCLK) during which the data bus is kept in high-Z state after a write access has started</td>
<td>Write</td>
<td>AHB clock cycle (HCLK)</td>
<td>1</td>
<td>255</td>
</tr>
</tbody>
</table>

### 22.7.1 External memory interface signals

The following tables list the signals that are typically used to interface NAND flash memory.

*Note:* The prefix “N” identifies the signals which are active low.

#### 8-bit NAND flash memory

**Table 197. 8-bit NAND flash**

<table>
<thead>
<tr>
<th>FMC signal name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A[17]</td>
<td>O</td>
<td>NAND flash address latch enable (ALE) signal</td>
</tr>
<tr>
<td>A[16]</td>
<td>O</td>
<td>NAND flash command latch enable (CLE) signal</td>
</tr>
<tr>
<td>D[7:0]</td>
<td>I/O</td>
<td>8-bit multiplexed, bidirectional address/data bus</td>
</tr>
</tbody>
</table>
Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

16-bit NAND flash memory

Table 198. 16-bit NAND flash

<table>
<thead>
<tr>
<th>FMC signal name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A[17]</td>
<td>O</td>
<td>NAND flash address latch enable (ALE) signal</td>
</tr>
<tr>
<td>A[16]</td>
<td>O</td>
<td>NAND flash command latch enable (CLE) signal</td>
</tr>
<tr>
<td>D[15:0]</td>
<td>I/O</td>
<td>16-bit multiplexed, bidirectional address/data bus</td>
</tr>
<tr>
<td>NCE</td>
<td>O</td>
<td>Chip select</td>
</tr>
<tr>
<td>NOE (= NRE)</td>
<td>O</td>
<td>Output enable (memory signal name: read enable, NRE)</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NWAIT/INT</td>
<td>I</td>
<td>NAND flash ready/busy input signal to the FMC</td>
</tr>
</tbody>
</table>

Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

22.7.2 NAND flash supported memories and transactions

Table 199 shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND flash controller are shown in gray.

Table 199. Supported memories and transactions

<table>
<thead>
<tr>
<th>Device</th>
<th>Mode</th>
<th>R/W</th>
<th>AHB data size</th>
<th>Memory data size</th>
<th>Allowed/not allowed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAND 8-bit</td>
<td>Asynchronous</td>
<td>R</td>
<td>8</td>
<td>8</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>8</td>
<td>8</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>16</td>
<td>8</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>16</td>
<td>8</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>32</td>
<td>8</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>32</td>
<td>8</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
</tbody>
</table>
22.7.3 Timing diagrams for NAND flash memory

The NAND flash memory bank is managed through a set of registers:
- Control register: FMC_PCR
- Interrupt status register: FMC_SR
- ECC register: FMC_ECCR
- Timing register for Common memory space: FMC_PMEM
- Timing register for Attribute memory space: FMC_PATT

Each timing configuration register contains three parameters used to define number of HCLK cycles for the three phases of any NAND flash access, plus one parameter that defines the timing for starting driving the data bus when a write access is performed. Figure 126 shows the timing parameter definitions for common memory accesses, knowing that Attribute memory space access timings are similar.

Table 199. Supported memories and transactions (continued)

<table>
<thead>
<tr>
<th>Device</th>
<th>Mode</th>
<th>R/W</th>
<th>AHB data size</th>
<th>Memory data size</th>
<th>Allowed/ not allowed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAND 16-bit</td>
<td>Asynchronous</td>
<td>R</td>
<td>8</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>NAND 16-bit</td>
<td>Asynchronous</td>
<td>W</td>
<td>8</td>
<td>16</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>NAND 16-bit</td>
<td>Asynchronous</td>
<td>R</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>NAND 16-bit</td>
<td>Asynchronous</td>
<td>W</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>NAND 16-bit</td>
<td>Asynchronous</td>
<td>R</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td>NAND 16-bit</td>
<td>Asynchronous</td>
<td>W</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
</tbody>
</table>

1. NOE remains high (inactive) during write accesses. NWE remains high (inactive) during read accesses.
2. For write access, the hold phase delay is (MEMHOLD) HCLK cycles and for read access is (MEMHOLD + 2) HCLK cycles.

Figure 126. NAND flash controller waveforms for common memory access
22.7.4 NAND flash operations

The command latch enable (CLE) and address latch enable (ALE) signals of the NAND flash memory device are driven by address signals from the FMC controller. This means that to send a command or an address to the NAND flash memory, the CPU has to perform a write to a specific address in its memory space.

A typical page read operation from the NAND flash device requires the following steps:

1. Program and enable the corresponding memory bank by configuring the FMC_PCR and FMC_PMEM (and for some devices, FMC_PATT, see Section 22.7.5: NAND flash prewait functionality) registers according to the characteristics of the NAND flash memory (PWID bits for the data bus width of the NAND flash, PTYP = 1, PWAITEN = 0 or 1 as needed, see Section 22.5.2: NAND flash memory address mapping for timing configuration).

2. The CPU performs a byte write to the common memory space, with data byte equal to one flash command byte (for example 0x00 for Samsung NAND flash devices). The LE input of the NAND flash memory is active during the write strobe (low pulse on NWE), thus the written byte is interpreted as a command by the NAND flash memory. Once the command is latched by the memory device, it does not need to be written again for the following page read operations.

3. The CPU can send the start address (STARTAD) for a read operation by writing four bytes (or three for smaller capacity devices), STARTAD[7:0], STARTAD[16:9], STARTAD[24:17] and finally STARTAD[25] (for 64 Mb x 8 bit NAND flash memories) in the common memory or attribute space. The ALE input of the NAND flash device is active during the write strobe (low pulse on NWE), thus the written bytes are interpreted as the start address for read operations. Using the attribute memory space makes it possible to use a different timing configuration of the FMC, which can be used to implement the prewait functionality needed by some NAND flash memories (see details in Section 22.7.5: NAND flash prewait functionality).

4. The controller waits for the NAND flash memory to be ready (R/NB signal high), before starting a new access to the same or another memory bank. While waiting, the controller holds the NCE signal active (low).

5. The CPU can then perform byte read operations from the common memory space to read the NAND flash page (data field + Spare field) byte by byte.

6. The next NAND flash page can be read without any CPU command or address write operation. This can be done in three different ways:
   – by simply performing the operation described in step 5
   – a new random address can be accessed by restarting the operation at step 3
   – a new command can be sent to the NAND flash device by restarting at step 2

22.7.5 NAND flash prewait functionality

Some NAND flash devices require that, after writing the last part of the address, the controller waits for the R/NB signal to go low. (see Figure 127).
1. CPU wrote byte 0x00 at address 0x7001 0000.
2. CPU wrote byte A7–A0 at address 0x7002 0000.
3. CPU wrote byte A16–A9 at address 0x7002 0000.
4. CPU wrote byte A24–A17 at address 0x7002 0000.
5. CPU wrote byte A25 at address 0x7802 0000: FMC performs a write access using FMC_PATT timing definition, where ATTHOLD ≥ 7 (providing that (7+1) × HCLK = 112 ns > tWB max). This guarantees that NCE remains low until R/NB goes low and high again (only requested for NAND flash memories where NCE is not don't care).

When this functionality is required, it can be ensured by programming the MEMHOLD value to meet the tWB timing. However any CPU read access to the NAND flash memory has a hold delay of (MEMHOLD + 2) HCLK cycles and CPU write access has a hold delay of (MEMHOLD) HCLK cycles inserted between the rising edge of the NWE signal and the next access.

To cope with this timing constraint, the attribute memory space can be used by programming its timing register with an ATTHOLD value that meets the tWB timing, and by keeping the MEMHOLD value at its minimum value. The CPU must then use the common memory space for all NAND flash read and write accesses, except when writing the last address byte to the NAND flash device, where the CPU must write to the attribute memory space.

22.7.6 Computation of the error correction code (ECC) in NAND flash memory

The FMC NAND Card controller includes two error correction code computation hardware blocks, one per memory bank. They reduce the host CPU workload when processing the ECC by software.
These two ECC blocks are identical and associated with Bank 2 and Bank 3. As a consequence, no hardware ECC computation is available for memories connected to Bank 4.

The ECC algorithm implemented in the FMC can perform 1-bit error correction and 2-bit error detection per 256, 512, 1024, 2048, 4096 or 8192 bytes read or written from/to the NAND flash memory. It is based on the Hamming coding algorithm and consists in calculating the row and column parity.

The ECC modules monitor the NAND flash data bus and read/write signals (NCE and NWE) each time the NAND flash memory bank is active.

The ECC operates as follows:
- When accessing NAND flash memory bank 2 or bank 3, the data present on the D[15:0] bus is latched and used for ECC computation.
- When accessing any other address in NAND flash memory, the ECC logic is idle, and does not perform any operation. As a result, write operations to define commands or addresses to the NAND flash memory are not taken into account for ECC computation.

Once the desired number of bytes has been read/written from/to the NAND flash memory by the host CPU, the FMC_ECCR registers must be read to retrieve the computed value. Once read, they must be cleared by resetting the ECCEN bit to '0'. To compute a new data block, the ECCEN bit must be set to one in the FMC_PCR registers.

To perform an ECC computation:
1. Enable the ECCEN bit in the FMC_PCR register.
2. Write data to the NAND flash memory page. While the NAND page is written, the ECC block computes the ECC value.
3. Read the ECC value available in the FMC_ECCR register and store it in a variable.
4. Clear the ECCEN bit and then enable it in the FMC_PCR register before reading back the written data from the NAND page. While the NAND page is read, the ECC block computes the ECC value.
5. Read the new ECC value available in the FMC_ECCR register.
6. If the two ECC values are the same, no correction is required, otherwise there is an ECC error and the software correction routine returns information on whether the error can be corrected or not.

### 22.7.7 NAND flash controller registers

#### NAND flash control registers (FMC_PCR)

Address offset: 0x80

Reset value: 0x0000 0018

<table>
<thead>
<tr>
<th>bit 31-0</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECCPS[2:0]</td>
<td>read/write</td>
</tr>
<tr>
<td>TAR3[2:0]</td>
<td>read/write</td>
</tr>
<tr>
<td>TAR[2:0]</td>
<td>read/write</td>
</tr>
<tr>
<td>TCLR[3:0]</td>
<td>read/write</td>
</tr>
<tr>
<td>Res.</td>
<td>read/write</td>
</tr>
<tr>
<td>Res.</td>
<td>read/write</td>
</tr>
<tr>
<td>ECCEN</td>
<td>read/write</td>
</tr>
<tr>
<td>PWID[1:0]</td>
<td>read/write</td>
</tr>
<tr>
<td>PTYP</td>
<td>read/write</td>
</tr>
<tr>
<td>PBKEN</td>
<td>read/write</td>
</tr>
<tr>
<td>PWAITEN</td>
<td>read/write</td>
</tr>
<tr>
<td>Res.</td>
<td>read/write</td>
</tr>
</tbody>
</table>
Flexible static memory controller (FMC)

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:17 **ECCPS[2:0]**: ECC page size
Defines the page size for the extended ECC:
- 000: 256 bytes
- 001: 512 bytes
- 010: 1024 bytes
- 011: 2048 bytes
- 100: 4096 bytes
- 101: 8192 bytes

Bits 16:13 **TAR[3:0]**: ALE to RE delay
Sets time from ALE low to RE low in number of AHB clock cycles (HCLK).
Time is: \( t_{ar} = (TAR + SET + 2) \times THCLK \) where THCLK is the HCLK clock period
- 0000: 1 HCLK cycle (default)
- 1111: 16 HCLK cycles

*Note: SET is MEMSET or ATTSET according to the addressed space.*

Bits 12:9 **TCLR[3:0]**: CLE to RE delay
Sets time from CLE low to RE low in number of AHB clock cycles (HCLK).
Time is \( t_{clr} = (TCLR + SET + 2) \times THCLK \) where THCLK is the HCLK clock period
- 0000: 1 HCLK cycle (default)
- 1111: 16 HCLK cycles

*Note: SET is MEMSET or ATTSET according to the addressed space.*

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **ECCEN**: ECC computation logic enable bit
- 0: ECC logic is disabled and reset (default after reset),
- 1: ECC logic is enabled.

Bits 5:4 **PWID[1:0]**: Data bus width
Defines the external memory device width.
- 00: 8 bits
- 01: 16 bits (default after reset).
- 10: reserved.
- 11: reserved.

Bit 3 **PTYP**: Memory type
Defines the type of device attached to the corresponding memory bank:
- 0: Reserved, must be kept at reset value
- 1: NAND flash (default after reset)

Bit 2 **PBKEN**: NAND flash memory bank enable bit
Enables the memory bank. Accessing a disabled memory bank causes an ERROR on AHB bus
- 0: Corresponding memory bank is disabled (default after reset)
- 1: Corresponding memory bank is enabled

Bit 1 **PWAITEN**: Wait feature enable bit
Enables the Wait feature for the NAND flash memory bank:
- 0: disabled
- 1: enabled

Bit 0 Reserved, must be kept at reset value.
FIFO status and interrupt register (FMC_SR)

Address offset: 0x84
Reset value: 0x0000 0040

This register contains information about the FIFO status and interrupt. The FMC features a FIFO that is used when writing to memories to transfer up to 16 words of data from the AHB.

This is used to quickly write to the FIFO and free the AHB for transactions to peripherals other than the FMC, while the FMC is draining its FIFO into the memory. One of these register bits indicates the status of the FIFO, for ECC purposes.

The ECC is calculated while the data are written to the memory. To read the correct ECC, the software must consequently wait until the FIFO is empty.

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

Bits 31:7 Reserved, must be kept at reset value.

- **Bit 6 FEMPT**: FIFO empty
  - Read-only bit that provides the status of the FIFO
  - 0: FIFO not empty
  - 1: FIFO empty

- **Bit 5 IFEN**: Interrupt falling edge detection enable bit
  - 0: Interrupt falling edge detection request disabled
  - 1: Interrupt falling edge detection request enabled

- **Bit 4 ILEN**: Interrupt high-level detection enable bit
  - 0: Interrupt high-level detection request disabled
  - 1: Interrupt high-level detection request enabled

- **Bit 3 IREN**: Interrupt rising edge detection enable bit
  - 0: Interrupt rising edge detection request disabled
  - 1: Interrupt rising edge detection request enabled

- **Bit 2 IFS**: Interrupt falling edge status
  - The flag is set by hardware and reset by software.
  - 0: No interrupt falling edge occurred
  - 1: Interrupt falling edge occurred
  - *Note: If this bit is written by software to 1 it is set.*

- **Bit 1 ILS**: Interrupt high-level status
  - The flag is set by hardware and reset by software.
  - 0: No Interrupt high-level occurred
  - 1: Interrupt high-level occurred
**Bit 0  IRS:** Interrupt rising edge status
   The flag is set by hardware and reset by software.
   0: No interrupt rising edge occurred
   1: Interrupt rising edge occurred

*Note: If this bit is written by software to 1 it is set.*

---

**Common memory space timing register (FMC_PMEM)**

Address offset: Address: 0x88
Reset value: 0xFCFC FCFC
The FMC_PMEM read/write register contains the timing information for NAND flash memory bank. This information is used to access either the common memory space of the NAND flash for command, address write access and data read/write access.

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

**Bits 31:24  MEMHIZ[7:0]:** Common memory x data bus Hi-Z time
- Defines the number of HCLK clock cycles during which the data bus is kept Hi-Z after the start of a NAND flash write access to common memory space on socket. This is only valid for write transactions:
  - 0000 0000: 1 HCLK cycle
  - 1111 1110: 255 HCLK cycles
  - 1111 1111: reserved.

**Bits 23:16  MEMHOLD[7:0]:** Common memory hold time
- Defines the number of HCLK clock cycles for write access and HCLK (+2) clock cycles for read access during which the address is held (and data for write accesses) after the command is deasserted (NWE, NOE), for NAND flash read or write access to common memory space on socket x:
  - 0000 0000: reserved.
  - 0000 0001: 1 HCLK cycle for write access / 3 HCLK cycles for read access
  - 1111 1111: 254 HCLK cycles for write access / 256 HCLK cycles for read access
  - 1111 1111: reserved.

**Bits 15:8  MEMWAIT[7:0]:** Common memory wait time
- Defines the minimum number of HCLK (+1) clock cycles to assert the command (NWE, NOE), for NAND flash read or write access to common memory space on socket. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK:
  - 0000 0000: reserved
  - 0000 0001: 2HCLK cycles (+1 wait cycle introduced by deasserting NWAIT)
  - 1111 1110: 255 HCLK cycles (+1 wait cycle introduced by deasserting NWAIT)
  - 1111 1111: reserved.
Bits 7:0 **MEMSET[7:0]:** Common memory x setup time

Defines the number of HCLK (+1) clock cycles to set up the address before the command assertion (NWE, NOE), for NAND flash read or write access to common memory space on socket x:

- 0000 0000: 1 HCLK cycle
- 1111 1110: 255 HCLK cycles
- 1111 1111: reserved

**Attribute memory space timing register (FMC_PATT)**

Address offset: 0x8C

Reset value: 0xFCFC FCFC

The FMC_PATT read/write register contains the timing information for NAND flash memory bank. It is used for 8-bit accesses to the attribute memory space of the NAND flash for the last address write access if the timing must differ from that of previous accesses (for Ready/Busy management, refer to Section 22.7.5: NAND flash prewait functionality).

<table>
<thead>
<tr>
<th>Bit 31-24</th>
<th>ATTTHIZ[7:0]: Attribute memory data bus Hi-Z time</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 23-16</th>
<th>ATTTHOLD[7:0]: Attribute memory hold time</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15-8</th>
<th>ATTWAIT[7:0]: Attribute memory wait time</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:24 **ATTTHIZ[7:0]:** Attribute memory data bus Hi-Z time

Defines the number of HCLK clock cycles during which the data bus is kept in Hi-Z after the start of a NAND flash write access to attribute memory space on socket. Only valid for write transaction:

- 0000 0000: 0 HCLK cycle
- 1111 1110: 255 HCLK cycles
- 1111 1111: reserved.

Bits 23:16 **ATTTHOLD[7:0]:** Attribute memory hold time

Defines the number of HCLK clock cycles for write access and HCLK (+2) clock cycles for read access during which the address is held (and data for write access) after the command deassertion (NWE, NOE), for NAND flash read or write access to attribute memory space on socket:

- 0000 0000: reserved
- 0000 0001: 1 HCLK cycle for write access / 3 HCLK cycles for read access
- 1111 1110: 254 HCLK cycles for write access / 256 HCLK cycles for read access
- 1111 1111: reserved.

Bits 15:8 **ATTWAIT[7:0]:** Attribute memory wait time

Defines the minimum number of HCLK (+1) clock cycles to assert the command (NWE, NOE), for NAND flash read or write access to attribute memory space on socket x. The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK:

- 0000 0000: reserved
- 0000 0001: 2 HCLK cycles (+ wait cycle introduced by deassertion of NWAIT)
- 1111 1110: 255 HCLK cycles (+ wait cycle introduced by deasserting NWAIT)
- 1111 1111: reserved.
Bits 7:0  **ATTSET[7:0]:** Attribute memory setup time

Defines the number of HCLK (+1) clock cycles to set up address before the command assertion (NWE, NOE), for NAND flash read or write access to attribute memory space on socket:

- 0000 0000: 1 HCLK cycle
- 1111 1110: 255 HCLK cycles
- 1111 1111: reserved.

**ECC result registers (FMC_ECCR)**

Address offset: 0x94
Reset value: 0x0000 0000

This register contain the current error correction code value computed by the ECC computation modules of the FMC NAND controller. When the CPU reads the data from a NAND flash memory page at the correct address (refer to *Section 22.7.6: Computation of the error correction code (ECC) in NAND flash memory*), the data read/written from/to the NAND flash memory are processed automatically by the ECC computation module. When X bytes have been read (according to the ECCPS field in the FMC_PCR registers), the CPU must read the computed ECC value from the FMC_ECC registers. It then verifies if these computed parity data are the same as the parity value recorded in the spare area, to determine whether a page is valid, and, to correct it otherwise. The FMC_ECCR register must be cleared after being read by setting the ECCEN bit to 0. To compute a new data block, the ECCEN bit must be set to 1.

<table>
<thead>
<tr>
<th>ECC[16:0]</th>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0000</td>
<td>1 HCLK cycle</td>
</tr>
<tr>
<td>1111 1110</td>
<td>255 HCLK cycles</td>
</tr>
<tr>
<td>1111 1111</td>
<td>reserved</td>
</tr>
</tbody>
</table>

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

Bits 31:0  **ECC[31:0]:** ECC result

This field contains the value computed by the ECC computation logic. *Table 200* describes the contents of these bitfields.

<table>
<thead>
<tr>
<th>ECCPS[2:0]</th>
<th>Page size in bytes</th>
<th>ECC bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>256</td>
<td>ECC[21:0]</td>
</tr>
<tr>
<td>001</td>
<td>512</td>
<td>ECC[23:0]</td>
</tr>
<tr>
<td>010</td>
<td>1024</td>
<td>ECC[25:0]</td>
</tr>
<tr>
<td>011</td>
<td>2048</td>
<td>ECC[27:0]</td>
</tr>
<tr>
<td>100</td>
<td>4096</td>
<td>ECC[29:0]</td>
</tr>
<tr>
<td>101</td>
<td>8192</td>
<td>ECC[31:0]</td>
</tr>
</tbody>
</table>
22.8 SDRAM controller

22.8.1 SDRAM controller main features

The main features of the SDRAM controller are the following:

- Two SDRAM banks with independent configuration
- 8-bit, 16-bit data bus width
- 13-bits Address Row, 11-bits Address Column, 4 internal banks: 4x16Mx16bit (128 MB), 4x16Mx8bit (64 MB)
- Word, half-word, byte access
- Automatic row and bank boundary management
- Multibank ping-pong access
- Programmable timing parameters
- Automatic Refresh operation with programmable Refresh rate
- Self-refresh mode
- Power-down mode
- SDRAM power-up initialization by software
- CAS latency of 1,2,3
- Cacheable Read FIFO with depth of 6 lines x32-bit (6 x14-bit address tag)

22.8.2 SDRAM External memory interface signals

At startup, the SDRAM I/O pins used to interface the FMC SDRAM controller with the external SDRAM devices must configured by the user application. The SDRAM controller I/O pins which are not used by the application, can be used for other purposes.

<table>
<thead>
<tr>
<th>SDRAM signal</th>
<th>I/O type</th>
<th>Description</th>
<th>Alternate function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDCLK</td>
<td>O</td>
<td>SDRAM clock</td>
<td>-</td>
</tr>
<tr>
<td>SDCKE[1:0]</td>
<td>O</td>
<td>SDRAM Bank 1 Clock Enable</td>
<td>SDRAM Bank 2 Clock Enable</td>
</tr>
<tr>
<td>SDNE[1:0]</td>
<td>O</td>
<td>SDRAM Bank 1 Chip Enable</td>
<td>SDRAM Bank 2 Chip Enable</td>
</tr>
<tr>
<td>D[15:0]</td>
<td>I/O</td>
<td>Bidirectional data bus</td>
<td>FMC_D[15:0]</td>
</tr>
<tr>
<td>BA[1:0]</td>
<td>O</td>
<td>Bank Address</td>
<td>FMC_A[15:14]</td>
</tr>
<tr>
<td>NRAS</td>
<td>O</td>
<td>Row Address Strobe</td>
<td>-</td>
</tr>
<tr>
<td>NCAS</td>
<td>O</td>
<td>Column Address Strobe</td>
<td>-</td>
</tr>
<tr>
<td>SDNWE</td>
<td>O</td>
<td>Write Enable</td>
<td>-</td>
</tr>
<tr>
<td>NBL[1:0]</td>
<td>O</td>
<td>Output Byte Mask for write accesses</td>
<td>(memory signal name: DQM[1:0])</td>
</tr>
</tbody>
</table>
22.8.3 SDRAM controller functional description

All SDRAM controller outputs (signals, address and data) change on the falling edge of the memory clock (FMC_SDCLK).

SDRAM initialization

The initialization sequence is managed by software. If the two banks are used, the initialization sequence must be generated simultaneously to Bank 1 and Bank 2 by setting the Target Bank bits CTB1 and CTB2 in the FMC_SDCMR register:

1. Program the memory device features into the FMC_SDCRx register. The SDRAM clock frequency, RBURST and RPIPE must be programmed in the FMC_SDCR1 register.
2. Program the memory device timing into the FMC_SDTRx register. The TRP and TRC timings must be programmed in the FMC_SDTR1 register.
3. Set MODE bits to ‘001’ and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to start delivering the clock to the memory (SDCKE is driven high).
4. Wait during the prescribed delay period. Typical delay is around 100 μs (refer to the SDRAM datasheet for the required delay after power-up).
5. Set MODE bits to ‘010’ and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to issue a “Precharge All” command.
6. Set MODE bits to ‘011’, and configure the Target Bank bits (CTB1 and/or CTB2) as well as the number of consecutive Auto-refresh commands (NRFS) in the FMC_SDCMR register. Refer to the SDRAM datasheet for the number of Auto-refresh commands that should be issued. Typical number is 8.
7. Configure the MRD field according to the SDRAM device, set the MODE bits to ‘100’, and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to issue a “Load Mode Register” command in order to program the SDRAM device. In particular:
   a) the CAS latency must be selected following configured value in FMC_SDCR1/2 registers
   b) the Burst Length (BL) of 1 must be selected by configuring the M[2:0] bits to 000 in the mode register. Refer to SDRAM device datasheet.
If the Mode Register is not the same for both SDRAM banks, this step has to be repeated twice, once for each bank, and the Target Bank bits set accordingly.
8. Program the refresh rate in the FMC_SDRTR register
   The refresh rate corresponds to the delay between refresh cycles. Its value must be adapted to SDRAM devices.
9. For mobile SDRAM devices, to program the extended mode register it should be done once the SDRAM device is initialized: First, a dummy read access should be performed while BA1=1 and BA=0 (refer to SDRAM address mapping section for BA[1:0] address mapping) in order to select the extended mode register instead of the load mode register and then program the needed value.

At this stage the SDRAM device is ready to accept commands. If a system reset occurs during an ongoing SDRAM access, the data bus might still be driven by the SDRAM device. Therefore the SDRAM device must be first reinitialized after reset before issuing any new access by the NOR flash/PSRAM/SRAM or NAND flash controller.
Note: If two SDRAM devices are connected to the FMC, all the accesses performed at the same time to both devices by the Command Mode register (Load Mode Register command) are issued using the timing parameters configured for SDRAM Bank 1 (TMRD and TRAS timings) in the FMC_SDTR1 register.

SDRAM controller write cycle

The SDRAM controller accepts single and burst write requests and translates them into single memory accesses. In both cases, the SDRAM controller keeps track of the active row for each bank to be able to perform consecutive write accesses to different banks (Multibank ping-pong access).

Before performing any write access, the SDRAM bank write protection must be disabled by clearing the WP bit in the FMC_SDCRx register.

Figure 128. Burst write SDRAM access waveforms

The SDRAM controller always checks the next access.

- If the next access is in the same row or in another active row, the write operation is carried out,
- if the next access targets another row (not active), the SDRAM controller generates a precharge command, activates the new row and initiates a write command.

SDRAM controller read cycle

The SDRAM controller accepts single and burst read requests and translates them into single memory accesses. In both cases, the SDRAM controller keeps track of the active row in each bank to be able to perform consecutive read accesses in different banks (Multibank ping-pong access).
The FMC SDRAM controller features a Cacheable read FIFO (6 lines x 32 bits). It is used to store data read in advance during the CAS latency period and the RPIPE delay following the below formula. The RBURST bit must be set in the FMC_SDCR1 register to anticipate the next read access.

Number for anticipated data = CAS latency + 1 + (RPIPE delay)/2

Examples:
- CAS latency = 3, RPIPE delay = 0: Four data (not committed) are stored in the FIFO.
- CAS latency = 3, RPIPE delay = 2: Five data (not committed) are stored in the FIFO.

The read FIFO features a 14-bit address tag to each line to identify its content: 11 bits for the column address, 2 bits to select the internal bank and the active row, and 1 bit to select the SDRAM device.

When the end of the row is reached in advance during an AHB burst read, the data read in advance (not committed) are not stored in the read FIFO. For single read access, data are correctly stored in the FIFO.

Each time a read request occurs, the SDRAM controller checks:
- If the address matches one of the address tags, data are directly read from the FIFO and the corresponding address tag/line content is cleared and the remaining data in the FIFO are compacted to avoid empty lines.
- Otherwise, a new read command is issued to the memory and the FIFO is updated with new data. If the FIFO is full, the older data are lost.
During a write access or a Precharge command, the read FIFO is flushed and ready to be filled with new data.

After the first read request, if the current access was not performed to a row boundary, the SDRAM controller anticipates the next read access during the CAS latency period and the RPIPE delay (if configured). This is done by incrementing the memory address. The following condition must be met:

- RBURST control bit should be set to ‘1’ in the FMC_SDCR1 register.
The address management depends on the next AHB request:

- **Next AHB request is sequential (AHB Burst)**
  - In this case, the SDRAM controller increments the address.

- **Next AHB request is not sequential**
  - If the new read request targets the same row or another active row, the new address is passed to the memory and the master is stalled for the CAS latency period, waiting for the new data from memory.
  - If the new read request does not target an active row, the SDRAM controller generates a Precharge command, activates the new row, and initiates a read command.

If the RURST is reset, the read FIFO is not used.

**Row and bank boundary management**

When a read or write access crosses a row boundary, if the next read or write access is sequential and the current access was performed to a row boundary, the SDRAM controller executes the following operations:

1. Precharge of the active row,
2. Activation of the new row
3. Start of a read/write command.

At a row boundary, the automatic activation of the next row is supported for all columns and data bus width configurations.

If necessary, the SDRAM controller inserts additional clock cycles between the following commands:

- Between Precharge and Active commands to match TRP parameter (only if the next access is in a different row in the same bank),
- Between Active and Read commands to match the TRCD parameter.

These parameters are defined into the FMC_SDTRx register.

Refer to *Figure 128* and *Figure 129* for read and burst write access crossing a row boundary.
Figure 131. Read access crossing row boundary

Figure 132. Write access crossing row boundary
If the next access is sequential and the current access crosses a bank boundary, the SDRAM controller activates the first row in the next bank and initiates a new read/write command. Two cases are possible:

- If the current bank is not the last one, the active row in the new bank must be precharged. At a bank boundary, the automatic activation of the next row is supported for all rows/columns and data bus width configuration.
- If the current bank is the last one and the selected SDRAM device is connected to Bank 1, the automatic activation of the next row in device connected to SDRAM Bank2 is not supported. A PALL software command must be issued on Bank1 before any any access on Bank2.

**SDRAM controller refresh cycle**

The Auto-refresh command is used to refresh the SDRAM device content. The SDRAM controller periodically issues auto-refresh commands. An internal counter is loaded with the COUNT value in the register FMC_SDRTR. This value defines the number of memory clock cycles between the refresh cycles (refresh rate). When this counter reaches zero, an internal pulse is generated.

If a memory access is ongoing, the auto-refresh request is delayed. However, if the memory access and the auto-refresh requests are generated simultaneously, the auto-refresh request takes precedence.

If the memory access occurs during an auto-refresh operation, the request is buffered and processed when the auto-refresh is complete.

If a new auto-refresh request occurs while the previous one was not served, the RE (Refresh Error) bit is set in the Status register. An Interrupt is generated if it has been enabled (REIE = ‘1’).

If SDRAM lines are not in idle state (not all row are closed), the SDRAM controller generates a PALL (Precharge ALL) command before the auto-refresh.

If the Auto-refresh command is generated by the FMC_SDCMR Command Mode register (Mode bits = ‘011’), a PALL command (Mode bits = ‘010’) must be issued first.

### 22.8.4 Low-power modes

Two low-power modes are available:

- **Self-refresh mode**
  The auto-refresh cycles are performed by the SDRAM device itself to retain data without external clocking.

- **Power-down mode**
  The auto-refresh cycles are performed by the SDRAM controller.

**Self-refresh mode**

This mode is selected by setting the MODE bits to ‘101’ and by configuring the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register.

The SDRAM clock stops running after a TRAS delay and the internal refresh timer stops counting only if one of the following conditions is met:

- A Self-refresh command is issued to both devices
- One of the devices is not activated (SDRAM bank is not initialized).
Before entering Self-Refresh mode, the SDRAM controller automatically issues a PALL command.

If the Write data FIFO is not empty, all data are sent to the memory before activating the Self-refresh mode and the BUSY status flag remains set.

In Self-refresh mode, all SDRAM device inputs become don’t care except for SDCKE which remains low.

The SDRAM device must remain in Self-refresh mode for a minimum period of time of TRAS and can remain in Self-refresh mode for an indefinite period beyond that. To guarantee this minimum period, the BUSY status flag remains high after the Self-refresh activation during a TRAS delay.

As soon as an SDRAM device is selected, the SDRAM controller generates a sequence of commands to exit from Self-refresh mode. After the memory access, the selected device remains in Normal mode.

To exit from Self-refresh, the MODE bits must be set to ‘000’ (Normal mode) and the Target Bank bits (CTB1 and/or CTB2) must be configured in the FMC_SDCMR register.

**Figure 133. Self-refresh mode**

- **SDCLK**
- **SDCKE**
- **COMMAND**:
  - PRECHARGE
  - NOP
  - AUTO REFRESH
  - NOP or COMMAND INHERIT
  - AUTO REFRESH
- **DOM/ DOML/DOMU**
- **A0- A9**, **A11, A12**
- **A10**:
  - ALL BANKS
- **Data[15:0]**: Hi-Z
  - PRECHARGE
  - tRP
  - tXSR

Precharge all active banks
Enter Self-refresh mode
CLK stable prior to existing Self-refresh mode
Exit Self-refresh mode
(restart refresh timebase)
Power-down mode

This mode is selected by setting the MODE bits to ‘110’ and by configuring the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register.

If the Write data FIFO is not empty, all data are sent to the memory before activating the Power-down mode.

As soon as an SDRAM device is selected, the SDRAM controller exits from the Power-down mode. After the memory access, the selected SDRAM device remains in Normal mode.

During Power-down mode, all SDRAM device input and output buffers are deactivated except for the SDCKE which remains low.

The SDRAM device cannot remain in Power-down mode longer than the refresh period and cannot perform the Auto-refresh cycles by itself. Therefore, the SDRAM controller carries out the refresh operation by executing the operations below:

1. Exit from Power-down mode and drive the SDCKE high
2. Generate the PALL command only if a row was active during Power-down mode
3. Generate the auto-refresh command
4. Drive SDCKE low again to return to Power-down mode.

To exit from Power-down mode, the MODE bits must be set to ‘000’ (Normal mode) and the Target Bank bits (CTB1 and/or CTB2) must be configured in the FMC_SDCMR register.
22.8.5 SDRAM controller registers

SDRAM control registers 1,2 (FMC_SDCR1,2)

Address offset: 0x140+ 0x4* (x - 1), (x = 1,2)
Reset value: 0x0000 02D0

This register contains the control parameters for each SDRAM memory bank.

| Bits 31:15 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bits 14:13</th>
<th>RPIPE[1:0]: Read pipe</th>
</tr>
</thead>
<tbody>
<tr>
<td>00: No clock cycle delay</td>
<td></td>
</tr>
<tr>
<td>01: One clock cycle delay</td>
<td></td>
</tr>
<tr>
<td>10: Two clock cycle delay</td>
<td></td>
</tr>
<tr>
<td>11: reserved.</td>
<td></td>
</tr>
</tbody>
</table>

*Note: The corresponding bits in the FMC_SDCR2 register is read only.*

<table>
<thead>
<tr>
<th>Bit 12</th>
<th>RBURST: Burst read</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: single read requests are not managed as bursts</td>
<td></td>
</tr>
<tr>
<td>1: single read requests are always managed as bursts</td>
<td></td>
</tr>
</tbody>
</table>

*Note: The corresponding bit in the FMC_SDCR2 register is don’t care.*

<table>
<thead>
<tr>
<th>Bits 11:10</th>
<th>SDCLK[1:0]: SDRAM clock configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>00: SDCLK clock disabled</td>
<td></td>
</tr>
<tr>
<td>10: SDCLK period = 2 x HCLK periods</td>
<td></td>
</tr>
<tr>
<td>11: SDCLK period = 3 x HCLK periods</td>
<td></td>
</tr>
</tbody>
</table>

*Note: The corresponding bits in the FMC_SDCR2 register are don’t care.*

<table>
<thead>
<tr>
<th>Bit 9</th>
<th>WP: Write protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Write accesses allowed</td>
<td></td>
</tr>
<tr>
<td>1: Write accesses ignored</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 8:7</th>
<th>CAS[1:0]: CAS Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>00: reserved.</td>
<td></td>
</tr>
<tr>
<td>01: 1 cycle</td>
<td></td>
</tr>
<tr>
<td>10: 2 cycles</td>
<td></td>
</tr>
<tr>
<td>11: 3 cycles</td>
<td></td>
</tr>
</tbody>
</table>

*Note: The corresponding bits in the FMC_SDCR2 register are don’t care.*
Bit 6  **NB**: Number of internal banks
   This bit sets the number of internal banks.
   0: Two internal Banks
   1: Four internal Banks

Bits 5:4  **MWID[1:0]**: Memory data bus width.
   These bits define the memory device width.
   00: 8 bits
   01: 16 bits
   10: reserved
   11: reserved.

Bits 3:2  **NR[1:0]**: Number of row address bits
   These bits define the number of bits of a row address.
   00: 11 bit
   01: 12 bits
   10: 13 bits
   11: reserved.

Bits 1:0  **NC[1:0]**: Number of column address bits
   These bits define the number of bits of a column address.
   00: 8 bits
   01: 9 bits
   10: 10 bits
   11: 11 bits.

**Note:** Before modifying the RBURST or RPIPE settings or disabling the SDCLK clock, the user must first send a PALL command to make sure ongoing operations are complete.

**SDRAM timing registers 1,2 (FMC_SDTR1,2)**

Address offset: 0x148 + 0x4 * (x - 1), (x = 1,2)

Reset value: 0xFFFF FFFF

This register contains the timing parameters of each SDRAM bank

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

Bits 31:28  Reserved, must be kept at reset value.

Bits 27:24  **TRCD[3:0]**: Row to column delay
   These bits define the delay between the Activate command and a Read/Write command in number of memory clock cycles.
   0000: 1 cycle.
   0001: 2 cycles
   ....
   1111: 16 cycles
Bits 23:20 **TRP[3:0]: Row precharge delay**
These bits define the delay between a Precharge command and another command in number of memory clock cycles. The TRP timing is only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRP must be programmed with the timing of the slowest device.

0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

*Note: The corresponding bits in the FMC_SDTR2 register are don't care.*

Bits 19:16 **TWR[3:0]: Recovery delay**
These bits define the delay between a Write and a Precharge command in number of memory clock cycles.

0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

*Note: TWR must be programmed to match the write recovery time (tWR) defined in the SDRAM datasheet, and to guarantee that:

\[ TWR \geq TRAS - TRCD \text{ and } TWR \geq TRC - TRCD - TRP \]

Example: TRAS= 4 cycles, TRCD= 2 cycles. So, TWR >= 2 cycles. TWR must be programmed to 0x1.

If two SDRAM devices are used, the FMC_SDTR1 and FMC_SDTR2 must be programmed with the same TWR timing corresponding to the slowest SDRAM device.

If only one SDRAM device is used, the TWR timing must be kept at reset value (0xF) for the not used bank.

Bits 15:12 **TRC[3:0]: Row cycle delay**
These bits define the delay between the Refresh command and the Activate command, as well as the delay between two consecutive Refresh commands. It is expressed in number of memory clock cycles. The TRC timing is only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRC must be programmed with the timings of the slowest device.

0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

*Note: TRC must match the TRC and TRFC (Auto Refresh period) timings defined in the SDRAM device datasheet.*

*Note: The corresponding bits in the FMC_SDTR2 register are don't care.*

Bits 11:8 **TRAS[3:0]: Self refresh time**
These bits define the minimum Self-refresh period in number of memory clock cycles.

0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles
Flexible static memory controller (FMC) RM0481

Bits 7:4 **TXSR[3:0]:** Exit Self-refresh delay

These bits define the delay from releasing the Self-refresh command to issuing the Activate command in number of memory clock cycles.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

*Note:* If two SDRAM devices are used, the FMC_SDTR1 and FMC_SDTR2 must be programmed with the same TXSR timing corresponding to the slowest SDRAM device.

Bits 3:0 **TMRD[3:0]:** Load Mode Register to Active

These bits define the delay between a Load Mode Register command and an Active or Refresh command in number of memory clock cycles.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles

*Note:* If two SDRAM devices are connected, all the accesses performed simultaneously to both devices by the Command Mode register (Load Mode Register command) are issued using the timing parameters configured for Bank 1 (TMRD and TRAS timings) in the FMC_SDTR1 register.

The TRP and TRC timings are only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRP and TRC timings must be programmed with the timings of the slowest device.

**SDRAM Command Mode register (FMC_SDCMR)**

Address offset: 0x150
Reset value: 0x0000 0000

This register contains the command issued when the SDRAM device is accessed. This register is used to initialize the SDRAM device, and to activate the Self-refresh and the Power-down modes. As soon as the MODE field is written, the command will be issued only to one or to both SDRAM banks according to CTB1 and CTB2 command bits. This register is the same for both SDRAM banks.

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

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

Bits 21:9 **MRD[12:0]:** Mode Register definition

This 13-bit field defines the SDRAM Mode Register content. The Mode Register is programmed using the Load Mode Register command.
SDRAM refresh timer register (FMC_SDRTR)

Address offset: 0x154
Reset value: 0x0000 0000

This register sets the refresh rate in number of SDCLK clock cycles between the refresh cycles by configuring the Refresh Timer Count value.

\[
\text{Refresh rate} = (\text{COUNT} + 1) \times \text{SDRAM clock frequency}
\]

\[
\text{COUNT} = \frac{(\text{SDRAM refresh period} / \text{Number of rows}) - 20}{20}
\]

Example

\[
\text{Refresh rate} = 64 \text{ ms} / (8196 \text{ rows}) = 7.81 \mu\text{s}
\]

where 64 ms is the SDRAM refresh period.
7.81μs × 60MHz = 468.6

The refresh rate must be increased by 20 SDRAM clock cycles (as in the above example) to obtain a safe margin if an internal refresh request occurs when a read request has been accepted. It corresponds to a COUNT value of '0000111000000' (448).

This 13-bit field is loaded into a timer which is decremented using the SDRAM clock. This timer generates a refresh pulse when zero is reached. The COUNT value must be set at least to 41 SDRAM clock cycles.

As soon as the FMC_SDRTR register is programmed, the timer starts counting. If the value programmed in the register is '0', no refresh is carried out. This register must not be reprogrammed after the initialization procedure to avoid modifying the refresh rate.

Each time a refresh pulse is generated, this 13-bit COUNT field is reloaded into the counter.

If a memory access is in progress, the Auto-refresh request is delayed. However, if the memory access and Auto-refresh requests are generated simultaneously, the Auto-refresh takes precedence. If the memory access occurs during a refresh operation, the request is buffered to be processed when the refresh is complete.

This register is common to SDRAM bank 1 and bank 2.

---

### Bits 31:15

Reserved, must be kept at reset value.

**Bit 14** REIE: RES Interrupt Enable
0: Interrupt is disabled
1: An Interrupt is generated if RE = 1

**Bits 13:1** COUNT[12:0]: Refresh Timer Count
This 13-bit field defines the refresh rate of the SDRAM device. It is expressed in number of memory clock cycles. It must be set at least to 41 SDRAM clock cycles (0x29).

Refresh rate = (COUNT + 1) x SDRAM frequency clock
COUNT = (SDRAM refresh period / Number of rows) - 20

**Bit 0** CRE: Clear Refresh error flag
This bit is used to clear the Refresh Error Flag (RE) in the Status Register.
0: no effect
1: Refresh Error flag is cleared

**Note:** The programmed COUNT value must not be equal to the sum of the following timings: TWR+TRP+TRC+TRCD+4 memory clock cycles.
SDRAM status register (FMC_SDSR)

Address offset: 0x158
Reset value: 0x0000 0000

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

Bit 5 BUSY: Busy status
This bit defines the status of the SDRAM controller after a Command Mode request
0: SDRAM Controller is ready to accept a new request
1: SDRAM Controller is not ready to accept a new request

Bits 4:3 MODES2[1:0]: Status Mode for Bank 2
This bit defines the Status Mode of SDRAM Bank 2.
00: Normal Mode
01: Self-refresh mode
10: Power-down mode

Bits 2:1 MODES1[1:0]: Status Mode for Bank 1
This bit defines the Status Mode of SDRAM Bank 1.
00: Normal Mode
01: Self-refresh mode
10: Power-down mode

Bit 0 RE: Refresh error flag
0: No refresh error has been detected
1: A refresh error has been detected
An interrupt is generated if REIE = 1 and RE = 1

22.8.6 FMC register map

Table 202. FMC register map and reset values
### Table 202. FMC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10</td>
<td>FMC_BCR3</td>
<td>000000110001010010</td>
</tr>
<tr>
<td>0x16</td>
<td>FMC_BCR4</td>
<td>00000000001100010</td>
</tr>
<tr>
<td>0x04</td>
<td>FMC_BTR1</td>
<td>00001111111111111111111111111111</td>
</tr>
<tr>
<td>0x0C</td>
<td>FMC_BTR2</td>
<td>00001111111111111111111111111111</td>
</tr>
<tr>
<td>0x14</td>
<td>FMC_BTR3</td>
<td>00001111111111111111111111111111</td>
</tr>
<tr>
<td>0x1C</td>
<td>FMC_BTR4</td>
<td>00001111111111111111111111111111</td>
</tr>
<tr>
<td>0x20</td>
<td>FMC_PCSCNTR</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x104</td>
<td>FMC_BWTR1</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x10C</td>
<td>FMC_BWTR2</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x114</td>
<td>FMC_BWTR3</td>
<td>00000000000000000000000000000000</td>
</tr>
</tbody>
</table>
Refer to Section 2.3 on page 111 for the register boundary addresses.
23 Octo-SPI interface (OCTOSPI)

23.1 Introduction

The OCTOSPI supports most external serial memories such as serial PSRAMs, serial NAND and serial NOR flash memories, HyperRAM™ and HyperFlash™ memories, with the following functional modes:

- indirect mode: all the operations are performed using the OCTOSPI registers to preset commands, addresses, data, and transfer parameters.
- automatic status-polling mode: the external memory status register is periodically read and an interrupt can be generated in case of flag setting. This feature is only available in regular-command protocol.
- memory-mapped mode: the external memory is memory mapped and it is seen by the system as if it was an internal memory, supporting both read and write operations.

The OCTOSPI supports the following protocols with associated frame formats:

- the regular-command frame format with the command, address, alternate byte, dummy cycles, and data phase
- the HyperBus™ frame format

23.2 OCTOSPI main features

- Functional modes: indirect, automatic status-polling, and memory-mapped
- Read and write support in memory-mapped mode
- External (P)SRAM memory support
- Support for single, dual, quad, and octal communication
- Dual memory configuration, where eight bits can be sent/received simultaneously by accessing two quad memories in parallel
- SDR (single-data rate) and DTR (double-transfer rate) support
- Data strobe support
- Fully programmable opcode
- Fully programmable frame format
- Support wrapped-type access to memory in read direction
- HyperBus support
- Integrated FIFO for reception and transmission
- Asynchronous bus clock versus kernel clock support
- 8-, 16-, and 32-bit data accesses allowed
- DMA protocol support
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error
- AHB interface with transaction acceptance limited to one: the interface accepts the next transfer on AHB bus only once the previous is completed on memory side.
## 23.3 OCTOSPI implementation

<table>
<thead>
<tr>
<th>OCTOSPI feature</th>
<th>OCTOSPI1/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>HyperBus standard compliant</td>
<td>X</td>
</tr>
<tr>
<td>Xcella standard compliant</td>
<td>X</td>
</tr>
<tr>
<td>XSPI (JEDEC251ES) standard compliant</td>
<td>X</td>
</tr>
<tr>
<td>AMBA® AHB compliant data interface</td>
<td>X</td>
</tr>
<tr>
<td>Dual AHB interface</td>
<td>X</td>
</tr>
<tr>
<td>Asynchronous AHB clock versus kernel clock</td>
<td>X</td>
</tr>
<tr>
<td>Functional modes: indirect, automatic status-polling, and memory-mapped</td>
<td>X</td>
</tr>
<tr>
<td>Read and write support in memory-mapped mode</td>
<td>X</td>
</tr>
<tr>
<td>Dual-quad configuration</td>
<td>X</td>
</tr>
<tr>
<td>SDR (single-data rate) and DTR (double-transfer rate)</td>
<td>X</td>
</tr>
<tr>
<td>Data strobe (DS,DQS)</td>
<td>X</td>
</tr>
<tr>
<td>Fully programmable opcode</td>
<td>X</td>
</tr>
<tr>
<td>Fully programmable frame format</td>
<td>X</td>
</tr>
<tr>
<td>Integrated FIFO for reception and transmission</td>
<td>X</td>
</tr>
<tr>
<td>8-, 16-, and 32-bit data accesses</td>
<td>X</td>
</tr>
<tr>
<td>Interrupt on FIFO threshold, timeout, operation complete, and access error</td>
<td>X</td>
</tr>
<tr>
<td>Extended CSHT timeout</td>
<td>X</td>
</tr>
<tr>
<td>Memory-mapped write</td>
<td>X</td>
</tr>
<tr>
<td>Refresh counter</td>
<td>X</td>
</tr>
<tr>
<td>GPDMA interface</td>
<td>X</td>
</tr>
<tr>
<td>Prefetch disable</td>
<td></td>
</tr>
<tr>
<td>Prefetch hardware software</td>
<td></td>
</tr>
</tbody>
</table>
23.4 OCTOSPI functional description

23.4.1 OCTOSPI block diagram

Figure 135. OCTOSPI block diagram in octal configuration

Figure 136. OCTOSPI block diagram in quad configuration
23.4.2 OCTOSPI pins and internal signals

Table 204. OCTOSPI input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCTOSPI_NCLK</td>
<td>Output</td>
<td>OCTOSPI inverted clock to support 1.8 V HyperBus protocol</td>
</tr>
<tr>
<td>OCTOSPI_CLK</td>
<td>Output</td>
<td>OCTOSPI clock</td>
</tr>
<tr>
<td>OCTOSPI_IO(n)</td>
<td>Input/output</td>
<td>OCTOSPI data pins</td>
</tr>
<tr>
<td>OCTOSPI_NCS</td>
<td>Output</td>
<td>Chip select for the memory</td>
</tr>
<tr>
<td>OCTOSPI_DQS</td>
<td>Input/output</td>
<td>Data strobe/write mask signal from/to the memory</td>
</tr>
</tbody>
</table>

Caution: Use the same configuration (output speed, HSLV) for all OCTOSPI input/output pins to avoid any data corruption.

Table 205. OCTOSPI internal signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>octospi_hclk</td>
<td>Input</td>
<td>OCTOSPI AHB clock</td>
</tr>
<tr>
<td>octospi_ker_ck</td>
<td>Input</td>
<td>OCTOSPI kernel clock</td>
</tr>
<tr>
<td>octospi_dma</td>
<td>N/A</td>
<td>DMA request signal</td>
</tr>
<tr>
<td>octospi_it</td>
<td>Output</td>
<td>Global interrupt line (see Table 208 for the multiple sources of interrupt)</td>
</tr>
</tbody>
</table>
23.4.3 OCTOSPI interface to memory modes

The OCTOSPI supports the following protocols:

- regular-command protocol
- HyperBus protocol

The OCTOSPI uses from 6 to 12 signals to interface with a memory, depending on the functional mode:

- NCS: chip-select
- CLK: communication clock
- NCLK: inverted clock used only in the 1.8 V HyperBus protocol
- DQS: data strobe used only in regular-command protocol as input only
- IO[3:0]: data bus LSB
- IO[7:4]:
  - data bus MSB used in dual-quad and octal configurations
  - data bus can be used as possible remap for quad-SPI mode

23.4.4 OCTOSPI regular-command protocol

When in regular-command protocol, the OCTOSPI communicates with the external device using commands. Each command can include the following phases:

- Instruction phase
- Address phase
- Alternate-byte phase
- Dummy-cycle phase
- Data phase

Any of these phases can be configured to be skipped but, in case of single-phase command, the only use case supported is instruction-phase-only.

The NCS falls before the start of each command and rises again after each command finishes.

In memory-mapped mode, both read and write operations are supported: as a consequence, some of the configuration registers are duplicated to specify write operations (read operations are configured using regular registers).

Figure 138. SDR read command in octal configuration

The specific regular-command protocol features are configured through the registers in the 0x0100-0x01FC offset range.
Instruction phase

During this phase, a 1- to 4-byte instruction is sent to the external device specifying the type of operation to be performed. The size of the instruction to be sent is configured by ISIZE[1:0] in OCTOSPI_CCR and the instruction is programmed in INSTRUCTION[31:0] of OCTOSPI_IR.

The instruction phase can optionally send:
- 1 bit at a time (over IO0, SO single in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode).

This can be configured using IMODE[2:0] of OCTOSPI_CCR.

The instruction can be sent in DTR mode on each rising and falling edge of the clock, by setting IDTR in OCTOSPI_CCR.

When IMODE[2:0] = 000 in OCTOSPI_CCR, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

In memory-mapped mode, the instruction used for the write operation is specified in OCTOSPI_WIR, and the instruction format is specified in OCTOSPI_WCCR. The instruction used for the read operation and the instruction format are specified in OCTOSPI_IR and OCTOSPI_CCR.

Address phase

In the address phase, 1 to 4 bytes are sent to the external device, to indicate the address of the operation. The number of address bytes to be sent is configured by ADSIZE[1:0] in OCTOSPI_CCR.

In indirect and automatic status-polling modes, the address bytes to be sent are specified by ADDRESS[31:0] in OCTOSPI_AR. In memory-mapped mode, the address is given directly via the AHB (from any master in the system).

The address phase can send:
- 1 bit at a time (over IO0, SO single in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ADMODE[2:0] in OCTOSPI_CCR.

The address can be sent in DTR mode (on each rising and falling edge of the clock) setting ADDTR in OCTOSPI_CCR.

When ADMODE[2:0] = 000, the address phase is skipped and the command sequence proceeds directly to the next phase, if any.

In memory-mapped mode, the address format for the write operation is specified in OCTOSPI_WCCR. The address format for the read operation is specified in OCTOSPI_CCR.
Alternate-byte phase

In the alternate-byte phase, 1 to 4 bytes are sent to the external device, generally to control the mode of operation. The number of alternate bytes to be sent is configured by ABSIZE[1:0] in OCTOSPI_CCR. The bytes to be sent are specified in OCTOSPI_ABR.

The alternate-byte phase can send:
- 1 bit at a time (over IO0, SO single in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ABMODE[2:0] in OCTOSPI_CCR.

The alternate bytes can be sent in DTR mode (on each rising and falling edge of the clock) setting ABDTR in OCTOSPI_CCR.

When ABMODE[2:0] = 000, the alternate-byte phase is skipped and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when the dual-SPI mode is used and only two cycles are used for the alternate bytes.

In this case, the firmware can use the quad-SPI mode (ABMODE[2:0] = 011) and send a byte with bits 7 and 3 of ALTERNATE[31:0] set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low), in OCTOSPI_IR.

The upper two bits of the nibble to be sent are then placed in bits 5:4 of ALTERNATE[31:0] while the lower two bits are placed in bits 1:0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE[31:0] must be set to 0x8A (1000_1010).

In memory-mapped mode, the alternate bytes used for the write operation are specified in OCTOSPI_WABR, and the alternate byte format is specified in OCTOSPI_WCCR. The alternate bytes used for read operation and the alternate byte format are specified in OCTOSPI_ABR and OCTOSPI_CCR.

Dummy-cycle phase (memory latency)

In the dummy-cycle phase, 1 to 31 cycles are given without any data being sent or received, in order to give the external device, the time to prepare for the data phase when the higher clock frequencies are used. The number of cycles given during this phase is specified by DCYC[4:0] in OCTOSPI_TCR. In both SDR and DTR modes, the duration is specified as a number of full CLK cycles.

When DCYC[4:0] = 00000, the dummy-cycle phase is skipped, and the command sequence proceeds directly to the data phase, if present.

In order to assure enough “turn-around” time for changing the data signals from the output mode to the input mode, there must be at least one dummy cycle when using the dual-SPI, the quad-SPI, or the octal-SPI mode, to receive data from the external device.

In memory-mapped mode, the dummy cycles for the write operations are specified in OCTOSPI_WTCR. The dummy cycles for the read operation are specified in OCTOSPI_TCR.
Data phase

During the data phase, any number of bytes can be sent to or received from the external device.

In indirect mode, the number of bytes to be sent/received is specified in OCTOSPI_DLR. In this mode, the data to be sent to the external device must be written to OCTOSPI_DR, while in indirect-read mode the data received from the external device is obtained by reading OCTOSPI_DR.

In automatic status-polling mode, the number of bytes to be received is specified in OCTOSPI_DLR, and the data received from the external device can be obtained by reading OCTOSPI_DR.

In memory-mapped mode, the data read or written, is sent or received directly over the AHB to the Cortex core or to a DMA.

The data phase can send/receive:
- 1 bit at a time (over IO0/IO1 (SO/SI respectively) in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using DMODE[2:0] in OCTOSPI_CCR.

The data can be sent or received in DTR mode (on each rising and falling edge of the clock) setting DDTR in OCTOSPI_CCR.

When DMODE[2:0] = 000, the data phase is skipped, and the command sequence finishes immediately by raising the NCS. This configuration must be used only in indirect-write mode.

In memory-mapped mode, the data format for the write operation is specified in OCTOSPI_WCCR. The data format for the read operation is specified in OCTOSPI_CCR.

DQS use

The DQS signal can be used for data strobing during the read transactions when the device toggles the DQS aligned with the data.

The DQS management can be enabled by setting DQSE in OCTOSPI_CCR.

---

**Figure 139. DTR read in octal-SPI mode with DQS (Macronix mode) example**

![Diagram showing DTR read in octal-SPI mode with DQS (Macronix mode)]
23.4.5 OCTOSPI regular-command protocol signal interface

Single-SPI mode

The legacy SPI mode allows just a single bit to be sent/received serially. In this mode, the data is sent to the external device over the SO signal (whose I/Os are shared with IO0). The data received from the external device arrives via SI (whose I/Os are shared with IO1).

The different phases can each be configured separately to use this single-SPI mode by setting to 001 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in single-SPI mode:
- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0 (to deactivate the “write protect” function).
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).
- IO4 to IO7 are in output mode and forced to 0.

This is the case even for the dummy phase if DMODE[2:0] = 001.

Dual-SPI mode

In dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual-SPI mode by setting to 010 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in dual-SPI mode:
- IO0/IO1 are at high-impedance (input) during the data phase for the read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase when DMODE[2:0] = 010, IO0/IO1 are always high-impedance.

Quad-SPI mode

In quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use the quad-SPI mode by setting to 011 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in quad-SPI mode:
- IO0 to IO3 are all at high-impedance (inputs) during the data phase for the read operations, and outputs in all other cases.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase when DMODE[2:0] = 011, IO0 to IO3 are all high-impedance.
Octal-SPI mode

In regular octal-SPI mode, the eight bits are sent/received simultaneously over the IO[0:7] signals.

The different phases can each be configured separately to use the octal-SPI mode by setting to 100 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase that is configured in octal-SPI mode, IO[0:7] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase when DMODE[2:0] = 100, IO[0:7] are all high-impedance.

Single-data rate (SDR) mode

By default, all the phases operate in SDR mode.

In this mode, when the OCTOSPI drives the IO0/SO, IO1 to IO7 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the OCTOSPI assumes that the external devices also send the data using CLK falling edge. By default (when SSHIFT = 0 in OCTOSPI_TCR), the signals are sampled using the following (rising) edge of CLK.

**Figure 140. SDR write command in octo-SPI mode example**

![Figure 140](https://example.com/image.png)

**Note:** Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

Double-transfer rate (DTR) mode

Each of the instruction, address, alternate-byte, and data phases can be configured to operate in DTR mode setting IDTR, ADDTR, ABDTR, and DDTR in OCTOSPI_CCR.

In memory-mapped mode, the DTR mode for each phase of the write operations is specified in OCTOSPI_WCCR. The DTR mode for each phase of the read operations is specified in OCTOSPI_CCR.

In DTR mode, when the OCTOSPI drives the IO0/SO and IO1 to IO7 signals in the instruction, address, and alternate-byte phases, a bit is sent or received on each of the falling and rising edges of CLK.

When receiving data in DTR mode, the OCTOSPI assumes that the external devices also send the data using both CLK rising and falling edges. When DDTR = 1 in OCTOSPI_CCR, the software must clear SSHIFT in OCTOSPI_TCR. Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

In DTR mode, it is recommended to set DHQC of OCTOSPI_TCR, to shift the outputs by a quarter of cycle and avoid holding issues on the memory side.
**Note:** DHQC must not be set when the prescaler value is 0, as this action leads to unpredictable behavior.

**Figure 141. DTR write in octal-SPI mode (Macronix mode) example**

![Diagram showing DTR write in octal-SPI mode](image)

**Note:** Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

**Dual-quad configuration**

When DMM = 1 in OCTOSPI_CR, the OCTOSPI is in dual-memory configuration: if DMODE = 011, two external Quad-SPI devices (device A and device B) are used in order to send/receive eight bits (or 16 bits in DTR mode) every cycle, effectively doubling the throughput.

Each device (A or B) uses the same CLK and NCS signals, but each has separate IO0 to IO3 signals.

The dual-quad configuration can be used in conjunction with the single-SPI, dual-SPI, and quad-SPI modes, as well as with either the SDR or DTR mode.

The device size, as specified by DEVSIZE[4:0] in OCTOSPI_DCR1, must reflect the total external device capacity that is the double of the size of one individual component.

If address X is even, then the byte that the OCTOSPI gives for address X is the byte at the address X/2 of device A, and the byte that the OCTOSPI gives for address X + 1 is the byte at the address X/2 of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.

When reading the status registers of the devices in dual-quad configuration, twice as many bytes must be read compared to the same read in regular-command protocol: if each device gives eight valid bits after the instruction for fetching the status register, then the OCTOSPI must be configured with a data length of 2 bytes (16 bits), and the OCTOSPI receives one byte from each device.

If each device gives a status of 16 bits, then the OCTOSPI must be configured to read 4 bytes to get all the status bits of both devices in dual-quad configuration. The least-significant byte of the result (in the data register) is the least-significant byte of device A status register. The next byte is the least-significant byte of device B status register. Then, the third byte of the data register is the device A second byte. The fourth byte is the device B second byte (if devices have 16-bit status registers).

An even number of bytes must always be accessed in dual-quad configuration. For this reason, bit 0 of DL[31:0] in OCTOSPI_DLR is stuck at 1 when DMM = 1.

In dual-quad configuration, the behavior of device A interface signals is basically the same as in normal mode. Device B interface signals have exactly the same waveforms as...
device A ones during the instruction, address, alternate-byte, and dummy-cycle phases. In other words, each device always receives the same instruction and the same address.

Then, during the data phase, the AIOx and the BIOx buses both transfer data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.

### 23.4.6 HyperBus protocol

The OCTOSPI can communicate with the external device using the HyperBus protocol. The HyperBus uses 11 to 12 pins depending on the operating voltage:

- IO[7:0] as bidirectional data bus
- RWDS for read and write data strobe and latency insertion (mapped on DQS pin)
- NCS
- CLK
- NCLK for 1.8 V operations (to support this mode, the device must be powered with 1.8 V)

The HyperBus does not require any command specification nor any alternate bytes. As a consequence, a separate register set is used to define the timing of the transaction.

The HyperBus frame is composed of the following phases:

- Command/address phase
- Data phase

The NCS falls before the start of a transaction and rises again after each transaction finishes.

**Figure 142. Example of HyperBus read operation**

![HyperBus read operation diagram](image)

**Note:** Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

The specific HyperBus features are configured through the registers in the 0x0200-0x02FC offset range.
Command/address phase

During this initial phase, the OCTOSPI sends 48 bits over IO[7:0] to specify the operations to be performed with the external device.

<table>
<thead>
<tr>
<th>CA bit</th>
<th>Bit name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>R/W#</td>
<td>Identifies the transaction as a read or a write.</td>
</tr>
<tr>
<td>46</td>
<td>Address space</td>
<td>Indicates if the transaction accesses the memory or the register space.</td>
</tr>
<tr>
<td>45</td>
<td>Burst type</td>
<td>Indicates if the burst is linear or wrapped.</td>
</tr>
<tr>
<td>44-16</td>
<td>Row and upper column address</td>
<td>Selects the row and the upper column addresses.</td>
</tr>
<tr>
<td>15-3</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td>2-0</td>
<td>Lower column address</td>
<td>Selects the starting 16-bit word within the half page.</td>
</tr>
</tbody>
</table>

The address space is configured through the memory type MTYP[2:0] in OCTOSPI_DCR1. The total size of the device is configured in DEVSIZE[4:0] of OCTOSPI_DCR1. In case of multi-chip product (MCP), the device size is the sum of all the sizes of all the MCP dies.

Read/write operation with initial latency

The HyperBus read and write operations need to respect two timings:
- \( t_{RWR} \): minimal read/write recovery time for the device (defined by TRWR[7:0] in OCTOSPI_HLCR)
- \( t_{ACC} \): access time for the device (defined by TACC[7:0] in OCTOSPI_HLCR) according to the memory latency

During the read operation, the RWDS is used by the device, in two ways (see Figure 142):
- during the command/address phase, to request an additional latency
- during the data phase, for data strobing

During the write operation, the RWDS is used:
- by the device, during the command/address phase, to request an additional latency.
- by the OCTOSPI, during the data phase, for write data masking.
Read/write operation with additional latency

If the device needs an additional latency (during refresh period of an SDRAM for example), RWDS must be tied to one during one of the RWDS signals, during the command/address phase.

An additional \( t_{\text{ACC}} \) duration is added by the OCTOSPI to meet the device request.

Figure 143. HyperBus write operation with initial latency

![Figure 143. HyperBus write operation with initial latency](image)

Figure 144. HyperBus read operation with additional latency

![Figure 144. HyperBus read operation with additional latency](image)
Fixed-latency mode

Some devices or some applications may not want to operate with a variable latency time as described above.

The latency can be forced to 2 x t\(_{\text{ACC}}\) by setting LM in OCTOSPI_HLCR.

In this OCTOSPI latency mode, the state of the RWDS signal is not taken into account by the OCTOSPI, and an additional latency is always added, leading to a fixed 2 x t\(_{\text{ACC}}\) latency time.

Write operation with no latency

Some devices can also require a zero latency for the write operations. This write-zero latency can be forced by setting WZL in OCTOSPI_HLCR.

Latency on page-crossing during the read operations

An additional latency can be needed by some devices for the read operation when crossing pages.

The initial latency must be respected for any page access, as a consequence, when the first access is close to the page boundary, a latency is automatically added at the page crossing to respect the t\(_{\text{ACC}}\) time.
23.4.7 Specific features

The OCTOSPI supports some specific features, such as:

- Wrap support
- NCS boundary and refresh

Wrap support

The OCTOSPI supports a hybrid wrap as defined by the HyperBus protocol. A hybrid wrap is also supported in the regular-command protocol.

In hybrid wrap, the transaction can continue after the initial wrap with an incremental access.

The wrap size supported by the target memory is configured by WRAPSIZE in OCTO_SPI_DCR2.

Wrap is supported only in memory-read direction and only for data size = 4 bytes. Wrapped reads are supported for both HyperBus and regular-command protocols. To enable wrapped-read accesses, the dedicated OCTO_SPI_WPxxx registers must be programmed according to the wrapped-read access characteristics. These registers apply for both HyperBus and regular-command protocols.

If the target memory is not supporting the hybrid wrap, WRAPSIZE must be set to 0.

Note: The wrap operation cannot be interrupted by a refresh. The refresh event is only considered after the wrap completion.

NCS boundary and refresh

Two processes can be activated to regulate the OCTOSPI transactions:

- NCS boundary
- Refresh

The NCS boundary feature limits a transaction to a boundary of aligned addresses. The size of the address to be aligned with, is configured by CSBOUND[4:0] in OCTO_SPI_DCR3: it is equal to $2^{CSBOUND}$.

As an example, if CSBOUND[4:0] = 0x4, the boundary is set to $2^4 = 16$ bytes. The NCS is then released each time the LSB address is equal to 0xF, and each time a new transaction is issued to address the next data.

If CSBOUND[4:0] = 0, the feature is disabled. A minimum value of three is recommended.
The NCS boundary feature cannot be used for flash memory devices in write mode since a command is necessary to program another page of the flash memory.

The refresh feature limits the duration of the transactions to the value programmed by REFRESH[31:0] in OCTOSPI_DCR4. The duration is expressed in number of cycles. This allows an external RAM to perform its internal refresh operation regularly.

The refresh value must be greater than the minimal transaction size in terms of number of cycles including the command/address/alternate/dummy phases.

If NCS boundary and refresh are enabled at the same time, the NCS is released on the first condition met.

**Restarting after an interrupted transfer**

When a read or write operation is interrupted by a timeout or communication regulation feature, the Octo-SPI interface, as soon as possible after getting back the port ownership, reissues the initial command sequence together with the address following the last address actually accessed before interruption. The transfer initially set goes on and ends seamlessly.

### 23.4.8 OCTOSPI operating mode introduction

The OCTOSPI has the following operating modes regardless of the low-level protocol used (either regular-command or HyperBus):

- indirect mode (read or write)
- automatic status-polling mode (only in regular-command protocol)
- memory-mapped mode

### 23.4.9 OCTOSPI indirect mode

In indirect mode, the commands are started by writing to the OCTOSPI registers, and data are transferred by writing or reading the data register, in a similar way to other communication peripherals.

When FMODE[1:0] = 00 in OCTOSPI_CR, the OCTOSPI is in indirect-write mode: bytes are sent to the external device during the data phase. Data are provided by writing to OCTOSPI_DR.

When FMODE[1:0] = 01, the OCTOSPI is in indirect-read mode: bytes are received from the external device during the data phase. Data are recovered by reading OCTOSPI_DR.

In indirect mode, when the OCTOSPI is configured in DTR mode over eight lanes with DQS disabled, the given starting address and the data length must be even.

**Note:** The OCTOSPI_AR register must be updated even if the start address is the same as the start address of the previous indirect access.

The number of bytes to be read/written is specified in OCTOSPI_DLR:

- If DL[31:0] = 0xFFFF FFFF, the data length is considered undefined and the OCTOSPI simply continues to transfer data until it reaches the end of the external device (as defined by DEVSIZE). If no bytes are to be transferred, DMODE[2:0] must be set to 0 in OCTOSPI_CCR.
- If DL[31:0] = 0xFFFF FFFFF and DEVSIZE[4:0] = 0x1F (its maximum value indicating at 4-Gbyte device), the transfers continue indefinitely, stopping only after an abort.
request or after the OCTOSPI is disabled. After the last memory address is read (at address 0xFFFF FFFF), reading continues with address = 0x0000 0000.

When the programmed number of bytes to be transmitted or received is reached, the TCF bit is set in OCTOSPI_SR, and an interrupt is generated if TCIE = 1 in OCTOSPI_CR. In the case of an undefined number of data, TCF is set when the limit of the external SPI memory is reached, according to the device size defined in OCTOSPI_DCR1.

**Triggering the start of a transfer in regular-command protocol**

Depending on the OCTOSPI configuration, there are three different ways to trigger the start of a transfer in indirect mode when using the regular-command protocol. In general, the start of transfer is triggered as soon as the software gives the last information that is necessary for the command. More specifically in indirect mode, a transfer starts when one of the following sequence of events occurs:

- if no address is necessary (ADMODE[2:0] = 000) and if no data need to be provided by the software (FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to INSTRUCTION[31:0] in OCTOSPI_IR
- if an address is necessary (when ADMODE[2:0] ≠ 000) and if no data need to be provided by the software (when FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to ADDRESS[31:0] in OCTOSPI_AR
- if data need to be provided by the software (when FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), and at the moment when a write is performed to DATA[31:0] in OCTOSPI_DR

A write to OCTOSPI_ABR never triggers the communication start. If alternate bytes are required, they must have been programmed before.

As soon as a command is started, the BUSY bit is automatically set in OCTOSPI_SR.

**Triggering the start of a transfer in HyperBus protocol**

Depending on the OCTOSPI configuration, there are different ways to trigger the start of a command in indirect mode. In general, it is triggered as soon as the firmware gives the last information that is necessary for the transfer to start, and more specifically, a communication in indirect mode is triggered by one of the following register settings, when it is the last one to be executed:

- when a write is performed to ADDRESS[31:0] (OCTOSPI_AR) with ADMODE[2:0] ≠ 000 in indirect read mode (FMODE[1:0] = 01).
- when a write is performed to DATA[31:0] (OCTOSPI_DR) in indirect-write mode (when FMODE = 00).
- when a (dummy) write is performed to INSTRUCTION[31:0] (OCTOSPI_IR) for indirect read mode (with ADMODE[2:0] = 000 and FMODE = 01).

As soon as a transfer is started, the BUSY bit (OCTOSPI_SR[5]) is automatically set.

**FIFO and data management**

Data in indirect mode passes through a 32-byte FIFO that is internal to the OCTOSPI. FLEVEL in OCTOSPI_SR indicates how many bytes are currently being held in the FIFO.

AHB burst transactions are supported. Data of the burst are successively written in OCTOSPI_DR, and immediately transferred in the internal FIFO.
In indirect-write mode (FMODE[1:0] = 00), the software adds data to the FIFO when it writes in OCTOSPI_DR. A word write adds 4 bytes to the FIFO, a half-word write adds 2 bytes, and a byte write adds only 1 byte. If the software adds too many bytes to the FIFO (more than indicated in DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

The byte/half-word accesses to OCTOSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES is used to define a FIFO threshold after which point the FIFO threshold flag, FTF, gets set. In indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there is any data left in the FIFO after the last byte is read from the external device, regardless of FTHRES setting. In indirect-write mode, the FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when the FTF is set. If DMAEN = 1, a DMA transfer is initiated when the FTF is set. The FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data has been transferred by the CPU or DMA).

The last data read in RX FIFO remains valid as long as there is no request for the next line. This means that, when the application reads several times in a row at the same location, the data is provided from the RX FIFO and not read again from the distant memory.

### 23.4.10 OCTOSPI automatic status-polling mode

In automatic status-polling mode, the OCTOSPI periodically starts a command to read a defined number of status bytes (up to four). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value. The automatic status-polling mode must be used only in regular-command protocol. For HyperBus protocol, it is not exploitable since the read status register into the HyperFlash memory must be performed in two steps (a write operation followed by a read operation).

The access to the device begins in the same manner as in indirect-read mode. BUSY in OCTOSPI_SR goes high at this point and stays high even between the periodic accesses.

The content of MASK[31:0] in OCTOSPI_PSMAR is used to mask the data from the external device in automatic status-polling mode:

- If the MASK[n] = 0, then bit n of the result is masked and not considered.
- If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] in OCTOSPI_PSMAR, then there is a match for bit n.

If PMM = 0 in OCTOSPI_CR, the AND-match mode is activated: SMF is set in OCTOSPI_SR only when there is a match on all of the unmasked bits.

If PMM = 1 in OCTOSPI_CR, the OR-match mode is activated: SMF gets set if there is a match on any of the unmasked bits.

An interrupt is called when SMF = 1 if SMIE = 1.

If APMS is set in OCTOSPI_CR, the operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at 1 and the periodic accesses continue until there is an abort or until the OCTOSPI is disabled (EN = 0).

OCTOSPI_DR contains the latest received status bytes (FIFO deactivated). The content of this register is not affected by the masking used in the matching logic. FTF in OCTOSPI_SR
is set as soon as a new reading of the status is complete. FTF is cleared as soon as the data is read.

In automatic status-polling mode, variable latency is not supported. The memory must then be configured in fixed latency.

### 23.4.11 OCTOSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

**Note:** *No more than 256 Mbytes can be addressed even if the external device capacity is larger.*

If an access is made to an address outside of the range defined by DEVSIZE[4:0] but still within the 256 Mbytes range, then an AHB error is given. The effect of this error depends on the AHB master that attempted the access:

- If it is the Cortex CPU, a hard-fault interrupt is generated.
- If it is a DMA, a DMA transfer error is generated, and the corresponding DMA channel is automatically disabled.

Byte, half-word, and word access types are all supported.

A support for execute in place (XIP) operation is implemented, where the OCTOSPI continues to load the bytes to the addresses following the most recent access. If subsequent accesses are continuous to the bytes that follow, then these operations end up quickly since their results were prefetched.

By default, the OCTOSPI never stops its prefetch operation. It either keeps the previous read operation active with the NCS maintained low or it relaunches a new transfer, even if no access to the external device occurs for a long time.

Since external devices tend to consume more when the NCS is held low, the application may want to activate the timeout counter (TCEN = 1 in OCTOSPI_CR): the NCS is released after a period defined by TIMEOUT[15:0] in OCTOSPI_LPTR, when x cycles have elapsed without access since the clock is inactive.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is an abort, or the peripheral is disabled.

It is not recommended to program the flash memory using the memory-mapped writes: the indirect-write mode fulfills this operation.

### 23.4.12 OCTOSPI configuration introduction

The OCTOSPI configuration is done in three steps:

1. OCTOSPI system configuration
2. OCTOSPI device configuration
3. OCTOSPI mode configuration

### 23.4.13 OCTOSPI system configuration

The OCTOSPI is configured using OCTOSPI_CR. The user must program:

- the functional mode with FMODE[1:0]
- the automatic status-polling mode behavior if needed with PMM and APMS
- the FIFO level with FTHRES
the DMA use with DMAEN
the timeout counter use with TCEN
the dual-memory configuration, if needed, with DMM

In case of an interrupt use, the respective enable bit can also be set during this phase.
If the timeout counter is used, the timeout value is programmed in OCTOSPI_LPTR.
The DMA channel must not be enabled during the OCTOSPI configuration: it must be enabled only when the operation is fully configured, to avoid any unexpected request generation.
The DMA and OCTOSPI must be configured in a coherent manner regarding data length: FTHRES value must reflect the DMA burst size.

23.4.14 OCTOSPI device configuration

The parameters related to the external device targeted are configured through OCTOSPI_DCR1 and OCTOSPI_DCR2. The user must program:

- the device size with DEVSIZE[4:0]
- the chip-select minimum high time with CSHT[5:0]
- the clock mode with FRCK and CKMODE
- the device frequency with PRESCALER[7:0]

MTYP[2:0] defines the memory type to be used for 8-line modes:
- Micron mode with D0/D1 ordering in 8-data-bit mode (DMODE[2:0] = 100)
- Macronix mode with D1/D0 ordering in 8-data-bit mode (DMODE[2:0] = 100).
  MTYP[2:0] = 001 targets Octaflash memory whereas MTYP[2:0] = 011 addresses OctaRAM™ memory having specific address phase (address is built with row and column to fit with Macronix requirements).
- HyperBus memory mode: the protocol follows the HyperBus specification.
- HyperBus register mode, addressing register space: the memory-mapped accesses in this mode must be noncacheable, or the indirect-read/write modes must be used.

DEVSIZE[4:0] defines the size of external memory using the following formula:

\[ \text{Number of bytes in the device} = 2^{\text{DEVSIZE}+1} \]

where DEVSIZE+1 is the number of address bits required to address the external device. The external device capacity can go up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

If DMM = 1, DEVSIZE[4:0] indicates the total capacity of the two devices together.

When the OCTOSPI executes two commands, one immediately after the other, it raises the chip-select signal (NCS) high between the two commands for only one CLK cycle by default.

If the external device requires more time between commands, the chip-select high time CSHT[5:0] can be used to specify the minimum number of CLK cycles for which the NCS must remain high.

CKMODE indicates the level that the CLK takes between commands (when NCS = 1).

In HyperBus protocol, the device timing (tACC and tRWR) and the latency mode must be configured in OCTOSPI_HLCR.
23.4.15 OCTOSPI regular-command mode configuration

Indirect mode configuration

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected, and data can be read from the external device.

When the OCTOSPI is used in indirect mode, the frames are constructed in the following way:
1. Specify a number of data bytes to read or write in OCTOSPI_DLR.
2. Specify the frame timing in OCTOSPI_TCR.
3. Specify the frame format in OCTOSPI_CCR.
4. Specify the instruction in OCTOSPI_IR.
5. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR.
6. Specify the targeted address in OCTOSPI_AR.
7. Enable the DMA channel if needed.
8. Read/write the data from/to the FIFO through OCTOSPI_DR (if no DMA usage).

If neither the address register (OCTOSPI_AR) nor the data register (OCTOSPI_DR) need to be updated for a particular command, then the command sequence starts as soon as OCTOSPI_IR is written. This is the case when both ADMODE[2:0] and DMODE[2:0] equal 000, or if just ADMODE[2:0] = 000 when in indirect-read mode (FMODE[1:0] = 01).

When an address is required (ADMODE[2:0] ≠ 000) and the data register does not need to be written (FMODE[1:0] = 01 or DMODE[2:0] = 000), the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

In case of data transmission (FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), the communication start is triggered by a write in the FIFO through OCTOSPI_DR.

Automatic status-polling mode configuration

The automatic status-polling mode is enabled by setting FMODE[1:0] = 10. In this mode, the programmed frame is sent and data are retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in OCTOSPI_DLR, it is ignored, and only 4 bytes are read. The periodicity is specified in OCTOSPI_PIR.

Once the status data has been retrieved, the following can be processed:
- Set SMF (an interrupt is generated if enabled).
- Stop automatically the periodic retrieving of the status bytes.

The received value can be masked with the value stored in OCTOSPI_PSMKR, and can be ORed or ANDed with the value stored in OCTOSPI_PSMAR.

In case of a match, SMF is set and an interrupt is generated if enabled. The OCTOSPI can be automatically stopped if AMPS is set. In any case, the latest retrieved value is available in OCTOSPI_DR.

When the OCTOSPI is used in automatic status-polling mode, the frames are constructed in the following way:
1. Specify the input mask in OCTOSPI_PSMKR.
2. Specify the comparison value in OCTOSPI_PSMAR.
3. Specify the read period in OCTOSPI_PIR.
4. Specify a number of data bytes to read in OCTOSPI_DLR.
5. Specify the frame timing in OCTOSPI_TCR.
6. Specify the frame format in OCTOSPI_CCR.
7. Specify the instruction in OCTOSPI_IR.
8. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR.
9. Specify the optional targeted address in OCTOSPI_AR.

If the address register (OCTOSPI_AR) does not need to be updated for a particular command, then the command sequence starts as soon as OCTOSPI_CCR is written. This is the case when ADMODE[2:0] = 000.

When an address is required (ADMODE[2:0] ≠ 000), the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

**Memory-mapped mode configuration**

In memory-mapped mode, the external device is seen as an internal memory but with some latency during accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using memory-mapped writes, as the internal flags for erase or programming status have to be polled. The indirect-write mode fulfills this operation, possibly in conjunction with the automatic status-polling mode.

The memory-mapped mode is entered by setting FMODE[1:0] = 11 in OCTOSPI_CR.

The programmed instruction and frame are sent when an AHB master accesses the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to OCTOSPI_DR in this mode returns zero.

The data length register (OCTOSPI_DLR) has no meaning in memory-mapped mode.

When the OCTOSPI is used in memory-mapped mode, the frames are constructed in the following way:
1. Specify the frame timing in OCTOSPI_TCR for read operation.
2. Specify the frame format in OCTOSPI_CCR for read operation.
3. Specify the instruction in OCTOSPI_IR.
4. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR for read operation.
5. Specify the frame timing in OCTOSPI_WTCR for write operation.
6. Specify the frame format in OCTOSPI_WCCR for write operation.
7. Specify the instruction in OCTOSPI_WIR.
8. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_WABR for write operation.

All configuration operations must be completed (ensured by checking BUSY = 0) before the first access to the memory area: any register write operation when BUSY = 1 has no effect and is not signaled with an error response. On the first access, the OCTOSPI becomes
busy, and no further configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit or to abort by setting the ABORT bit.

**OCTOSPI delayed data sampling when no DQS is used**

By default, when no DQS is used, the OCTOSPI samples the data driven by the external device one half of a CLK cycle after the external device drives the signal.

In case of any external signal delays, it may be useful to sample the data later. Using SSHIFT in OCTOSPI_TCR, the sampling of the data can be shifted by half of a CLK cycle.

The firmware must clear SSHIFT when the data phase is configured in DTR mode (DDTR = 1).

**OCTOSPI delayed data sampling when DQS is used**

When external DQS is used as a sampling clock, it can be shifted in time to compensate the data propagation delay. This shift is performed by an external delay block located outside the OCTOSPI. The control of this feature depends on the device implementation (see the product reference manual for more details).

In configurations where delay does not need to be compensated, the external delay block can be bypassed by setting DLYBYP in OCTOSPI_DCR1.

### 23.4.16 OCTOSPI HyperBus protocol configuration

**Indirect mode configuration (HyperBus)**

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected where data can be read from the external device. ADMODE must be configured with a value different from 000 (for instance ADMODE = 100).

When the OCTOSPI is used in indirect mode, the frames are constructed in the following way:
1. Specify a number of data bytes to read or write in OCTOSPI_DLR.
2. Specify the targeted address in OCTOSPI_AR.
3. Enable the DMA channel if needed.
4. Read/write the data from/to the FIFO through OCTOSPI_DR (if no DMA usage).

In indirect-read mode, the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

In indirect-write mode, the communication start is triggered by a write in the FIFO through OCTOSPI_DR.

**Memory-mapped mode configuration (HyperBus)**

In memory-mapped mode, the external device is seen as an internal memory but with some latency during the accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using the memory-mapped writes: the indirect-write mode fulfills this operation.

The memory-mapped mode is entered by setting FMODE[1:0] = 11. The programmed instruction and frame is sent when an AHB master accesses the memory-mapped space.
The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to OCTOSPI_DR in this mode returns zero.

The data length register (OCTOSPI_DLR) has no meaning in memory-mapped mode.

All the configuration operation must be completed before the first access to the memory area. On the first access, the OCTOSPI becomes busy, and no configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit, or to abort by setting the ABORT bit.

### 23.4.17 OCTOSPI error management

An error can be generated in the following cases:

- in indirect or automatic status-polling mode, when a wrong address has been programmed in OCTOSPI_AR (according to the device size defined by DEVSIZE[4:0]): this sets TEF and an interrupt is generated if enabled.
- in indirect mode, if the address plus the data length exceed the device size: TEF is set as soon as the access is triggered.
- in memory-mapped mode when an out-of-range access is done by an AHB master: this generates an AHB error as a response to the faulty AHB request.
- when the memory-mapped mode is disabled: an access to the memory-mapped area generates an AHB error as a response to the faulty AHB request.

The OCTOSPI generates an AHB slave error in the following situations:

- The memory-mapped mode is disabled and an AHB read request occurs.
- A read or write address exceeds the size of the external memory.
- An abort is received while a read or write burst is ongoing.
- The OCTOSPI is disabled while a read or write burst is ongoing.
- A write wrap burst is received.
- A write request is received while DQSE = 0 in OCTOSPI_WCCR in octal DTR mode or in dual-memory configuration.
- Write request is received while DMODE[2:0] = 000 (no data phase), except when MTYP[2:0] is HyperBus.
- Illegal access size when wrap read burst. This means that the HSIZE is different from 4 bytes (only for memory-mapped mode).
- Illegal wrap size when receiving read wrap burst with size different from 4 bytes (only for memory-mapped mode).

### 23.4.18 OCTOSPI BUSY and ABORT

Once the OCTOSPI starts an operation with the external device, BUSY is automatically set in OCTOSPI_SR.

In indirect mode, BUSY is reset once the OCTOSPI has completed the requested command sequence and the FIFO is empty.

In automatic status-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1 or due to an abort.

After the first access in memory-mapped mode, BUSY goes low only on an abort.

Any operation can be aborted by setting ABORT in OCTOSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.
Before setting ABORT, the software must ensure that all the current transactions are finished using the synchronization barriers. When DMA is enabled to handle the data read or write operations in OCTOSPI_DR, it is recommended to disable the DMA channel before aborting the OCTOSPI.

**Note:** Some devices may misbehave if a write operation to a status register is aborted.

### 23.4.19 OCTOSPI reconfiguration or deactivation

Before any OCTOSPI reconfiguration, the software must ensure that all the transactions are completed:

- After a memory-mapped write, the software must perform a dummy read followed by a synchronization barrier, then an abort.
- After a memory-mapped read, the software must perform a synchronization barrier than an abort.

### 23.4.20 NCS behavior

By default, NCS is high, deselecting the external device. NCS falls before an operation begins and rises as soon as it finishes.

When CKMODE = 0 (clock mode 0: CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

**Figure 148. NCS when CKMODE = 0 (T = CLK period)**

![Figure 148](image1)

When CKMODE = 1 (clock mode 3: CLK goes high when no operation is in progress) and when in SDR mode, NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

**Figure 149. NCS when CKMODE = 1 in SDR mode (T = CLK period)**

![Figure 149](image2)
When the CKMODE = 1 (clock mode 3) and DDTR = 1 (data DTR mode), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final active rising CLK edge (see the figure below). Because the DTR operations must finish with a falling edge, CLK is low when NCS rises, and CLK rises back up one half of a CLK cycle afterwards.

Figure 150. NCS when CKMODE = 1 in DTR mode (T = CLK period)

When the FIFO stays full during a read operation, or if the FIFO stays empty during a write operation, the operation stalls and CLK stays low until the software services the FIFO. If an abort occurs when an operation is stalled, NCS rises just after the abort is requested and then CLK rises one half of a CLK cycle later (see the figure below).

Figure 151. NCS when CKMODE = 1 with an abort (T = CLK period)

### 23.5 Address alignment and data number

The table below summarizes the effect of the address alignment and programmed data number depending on the use case.

**Table 207. Address alignment cases**

<table>
<thead>
<tr>
<th>Memory type</th>
<th>Transaction type</th>
<th>Constraint on address(1)</th>
<th>Impact if constraint on address not respected</th>
<th>Constraint on number of bytes(1)</th>
<th>Impact if constraint on bytes not respected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single, dual, quad flash or SRAM (DMM = 0)</td>
<td>IND(2) read</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>MM(3) read</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>IND write</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

(1) Depending on the use case.
### OCTOSPI interrupts

An interrupt can be produced on the following events:

- Timeout
- Status match
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available to provide more flexibility.

---

**Table 207. Address alignment cases (continued)**

<table>
<thead>
<tr>
<th>Memory type</th>
<th>Transaction type</th>
<th>Constraint on address(1)</th>
<th>Impact if constraint on address not respected</th>
<th>Constraint on number of bytes(1)</th>
<th>Impact if constraint on bytes not respected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single, dual, quad flash or SRAM (DMM = 1)</td>
<td>IND read</td>
<td>Even</td>
<td>ADDR[0] is set to 0.(4)</td>
<td>Even DLR[0] is set to 1.(5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM read</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>IND write</td>
<td>Even</td>
<td>ADDR[0] is set to 0.(4)</td>
<td>Even DLR[0] is set to 1.(5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td>Even</td>
<td>Slave error</td>
<td>Even</td>
<td>Last byte is lost.</td>
</tr>
<tr>
<td>Octal flash in SDR mode</td>
<td>IND read</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM read</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>IND write</td>
<td>Even</td>
<td>ADDR[0] is set to 0.(4)</td>
<td>Even DLR[0] is set to 1.(5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td>Even</td>
<td>Slave error</td>
<td>Even</td>
<td>Last byte is lost.</td>
</tr>
<tr>
<td>Octal memory in DTR mode without WDM(6)</td>
<td>IND read</td>
<td>Even</td>
<td>ADDR[0] is set to 0.(4)</td>
<td>Even DLR[0] is set to 1.(5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM read</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>IND write</td>
<td>Even</td>
<td>ADDR[0] is set to 0.(4)</td>
<td>Even DLR[0] is set to 1.(5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td>Even</td>
<td>Slave error</td>
<td>Even</td>
<td>Last byte is lost.</td>
</tr>
<tr>
<td>Octal flash or RAM in DTR mode with WDM</td>
<td>IND read</td>
<td>Even</td>
<td>ADDR[0] is set to 0.(4)</td>
<td>Even DLR[0] is set to 1.(5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM read</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IND write</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>HyperBus</td>
<td>IND read</td>
<td>Even</td>
<td>ADDR[0] is set to 0.(4)</td>
<td>Even DLR[0] is set to 1.(5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM read</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IND write</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

1. To be respected by the software.
2. IND = indirect mode.
3. MM = memory-mapped mode
4. Extra data at transfer start.
5. Extra data at transfer end.
6. WDM = write data mask.
23.7 OCTOSPI registers

23.7.1 OCTOSPI control register (OCTOSPI_CR)

Address offset: 0x0000
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>TOF</td>
<td>TOIE</td>
</tr>
<tr>
<td>Status match</td>
<td>SMF</td>
<td>SMIE</td>
</tr>
<tr>
<td>FIFO threshold</td>
<td>FTF</td>
<td>FTIE</td>
</tr>
<tr>
<td>Transfer complete</td>
<td>TCF</td>
<td>TCIE</td>
</tr>
<tr>
<td>Transfer error</td>
<td>TEF</td>
<td>TEIE</td>
</tr>
</tbody>
</table>

Table 208. OCTOSPI interrupt requests

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>TOF</td>
<td>TOIE</td>
</tr>
<tr>
<td>Status match</td>
<td>SMF</td>
<td>SMIE</td>
</tr>
<tr>
<td>FIFO threshold</td>
<td>FTF</td>
<td>FTIE</td>
</tr>
<tr>
<td>Transfer complete</td>
<td>TCF</td>
<td>TCIE</td>
</tr>
<tr>
<td>Transfer error</td>
<td>TEF</td>
<td>TEIE</td>
</tr>
</tbody>
</table>

Bits 31:30  Reserved, must be kept at reset value.

Bits 29:28  **FMODE[1:0]:** Functional mode

This bitfield defines the OCTOSPI functional mode of operation.

00: Indirect-write mode
01: Indirect-read mode
10: Automatic status-polling mode (relevant in regular-command protocol only)
11: Memory-mapped mode

If DMAEN = 1 already, then the DMA controller for the corresponding channel must be disabled before changing the FMODE[1:0] value. If FMODE[1:0] and FTHRES[4:0] are wrongly updated while DMAEN = 1, the DMA request signal automatically goes to inactive state.

*Note: This bitfield can be modified only when BUSY = 0.*

Bits 27:24  Reserved, must be kept at reset value.

Bit 23  **PMM:** Polling match mode

This bit indicates which method must be used to determine a match during the automatic status-polling mode.

0: AND-match mode, SMF is set if all the unmasked bits received from the device match the corresponding bits in the match register.
1: OR-match mode, SMF is set if any of the unmasked bits received from the device matches its corresponding bit in the match register.

*Note: This bit can be modified only when BUSY = 0.*
Bit 22  **APMS**: Automatic status-polling mode stop  
This bit determines if the automatic status-polling mode is stopped after a match.  
0: Automatic status-polling mode is stopped only by abort or by disabling the OCTOSPI.  
1: Automatic status-polling mode stops as soon as there is a match.  
*Note: This bit can be modified only when BUSY = 0.*

Bit 21  Reserved, must be kept at reset value.

Bit 20  **TOIE**: Timeout interrupt enable  
This bit enables the timeout interrupt.  
0: Interrupt disabled  
1: Interrupt enabled

Bit 19  **SMIE**: Status-match interrupt enable  
This bit enables the status-match interrupt.  
0: Interrupt disabled  
1: Interrupt enabled

Bit 18  **FTIE**: FIFO threshold interrupt enable  
This bit enables the FIFO threshold interrupt.  
0: Interrupt disabled  
1: Interrupt enabled

Bit 17  **TCIE**: Transfer complete interrupt enable  
This bit enables the transfer complete interrupt.  
0: Interrupt disabled  
1: Interrupt enabled

Bit 16  **TEIE**: Transfer error interrupt enable  
This bit enables the transfer error interrupt.  
0: Interrupt disabled  
1: Interrupt enabled

Bits 15:13  Reserved, must be kept at reset value.

Bits 12:8  **FTHRES[4:0]**: FIFO threshold level  
This bitfield defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag FTF in OCTOSPI_SR, to be set.  
00000: FTF is set if there are one or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are one or more valid bytes can be read from the FIFO in indirect-read mode.  
00001: FTF is set if there are two or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are two or more valid bytes can be read from the FIFO in indirect-read mode.  
...  
11111: FTF is set if there are 32 free bytes available to be written to in the FIFO in indirect-write mode, or if there are 32 valid bytes can be read from the FIFO in indirect-read mode.  
*Note: If DMAEN = 1, the DMA controller for the corresponding channel must be disabled before changing the FTHRES[4:0] value.*

Bit 7  **MSEL**: External memory select  
This bit selects the external memory to be addressed in single-, dual-, quad-SPI mode in single-memory configuration (when DMM = 0).  
0: External memory 1 selected (data exchanged over IO[3:0])  
1: External memory 2 selected (data exchanged over IO[7:4])  
This bit is ignored when DMM = 1 or when octal-SPI mode is selected.
Bit 6 **DMM**: Dual-memory configuration
This bit activates the dual-memory configuration, where two external devices are used simultaneously to double the throughput and the capacity
0: Dual-memory configuration disabled
1: Dual-memory configuration enabled
*Note: This bit can be modified only when BUSY = 0.*

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **TCEN**: Timeout counter enable
This bit is valid only when the memory-mapped mode (FMODE[1:0] = 11) is selected. This bit enables the timeout counter.
0: The timeout counter is disabled, and thus the chip-select (NCS) remains active indefinitely after an access in memory-mapped mode.
1: The timeout counter is enabled, and thus the chip-select is released in the memory-mapped mode after TIMEOUT[15:0] cycles of external device inactivity.
*Note: This bit can be modified only when BUSY = 0.*

Bit 2 **DMAEN**: DMA enable
In indirect mode, the DMA can be used to input or output data via OCTOSPI_DR. DMA transfers are initiated when FTF is set.
0: DMA disabled for indirect mode
1: DMA enabled for indirect mode
*Note: Resetting the DMAEN bit while a DMA transfer is ongoing, breaks the handshake with the DMA. Do not write this bit during DMA operation.*

Bit 1 **ABORT**: Abort request
This bit aborts the ongoing command sequence. It is automatically reset once the abort is completed. This bit stops the current transfer.
0: No abort requested
1: Abort requested
*Note: This bit is always read as 0.*

Bit 0 **EN**: Enable
This bit enables the OCTOSPI.
0: OCTOSPI disabled
1: OCTOSPI enabled
*Note: The DMA request can be aborted without having received the ACK in case this EN bit is cleared during the operation.
In case this bit is set to 0 during a DMA transfer, the REQ signal to DMA returns to inactive state without waiting for the ACK signal from DMA to be active.*
23.7.2 OCTOSPI device configuration register 1 (OCTOSPI_DCR1)

Address offset: 0x0008
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

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

Bits 26:24 MTYP[2:0]: Memory type
This bitfield indicates the type of memory to be supported.
000: Micron mode, D0/D1 ordering in DTR 8-data-bit mode. regular-command protocol in single-, dual-, quad- and octal-SPI modes.
Note: In this mode, DQS signal polarity is inverted with respect to the memory clock signal. This is the default value and care must be taken to change MTYP[2:0] for memories different from Micron.
001: Macronix mode, D1/D0 ordering in DTR 8-data-bit mode. regular-command protocol in single-, dual-, quad- and octal-SPI modes.
010: Standard mode
011: Macronix RAM mode, D1/D0 ordering in DTR 8-data-bit mode. regular-command protocol in single-, dual-, quad- and octal-SPI modes with dedicated address mapping.
100: HyperBus memory mode, the protocol follows the HyperBus specification.
101: HyperBus register mode, addressing register space. The memory-mapped accesses in this mode must be non-cacheable, or indirect-read/write modes must be used.
Others: Reserved

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 DEVSIZE[4:0]: Device size
This bitfield defines the size of the external device using the following formula:
Number of bytes in device = \(2^{\text{DEVSIZE} + 1}\).
DEVSIZE + 1 is effectively the number of address bits required to address the external device. The device capacity can be up to 4 Gbytes (addressed using 32-bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.
In regular-command protocol, if DMM = 1, DEVSIZE[4:0] indicates the capacity of one of the two external devices.

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

Bits 13:8 CSHT[5:0]: Chip-select high time
CSHT + 1 defines the minimum number of CLK cycles where the chip-select (NCS) must remain high between commands issued to the external device.
0x0: NCS stays high for at least 1 cycle between external device commands.
0x1: NCS stays high for at least 2 cycles between external device commands.
...
0x3F: NCS stays high for at least 64 cycles between external device commands.

Bits 7:4 Reserved, must be kept at reset value.
Bit 3 **DLYBYP**: Delay block bypass
- **0**: The internal sampling clock (called feedback clock) or the DQS data strobe external signal is delayed by the delay block (for more details on this block, refer to the dedicated section of the reference manual as it is not part of the OCTOSPI peripheral).
- **1**: The delay block is bypassed, so the internal sampling clock or the DQS data strobe external signal is not affected by the delay block. The delay is shorter than when the delay block is not bypassed, even with the delay value set to minimum value in delay block.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **FRCK**: Free running clock
- **0**: CLK is not free running.
- **1**: CLK is free running (always provided).

*Note*: Free running clock mode is intended for delay calibration only. No memory or other device access is possible when FRCK is set.

Bit 0 **CKMODE**: Clock mode 0/mode 3
- **0**: CLK must stay low while NCS is high (chip-select released). This is referred to as clock mode 0.
- **1**: CLK must stay high while NCS is high (chip-select released). This is referred to as clock mode 3.

## 23.7.3 OCTOSPI device configuration register 2 (OCTOSPI_DCR2)

Address offset: 0x000C

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **WRAPSIZE[2:0]**: Wrap size
- **000**: Wrapped reads are not supported by the memory.
- **010**: External memory supports wrap size of 16 bytes.
- **011**: External memory supports wrap size of 32 bytes.
- **100**: External memory supports wrap size of 64 bytes.
- **101**: External memory supports wrap size of 128 bytes.
- Others: Reserved

Bits 15:8 Reserved, must be kept at reset value.
23.7.4 OCTOSPI device configuration register 3 (OCTOSPI_DCR3)

Address offset: 0x0010
Reset value: 0x0000 0000
This register can be modified only when BUSY = 0.

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

Bits 31:21: Reserved, must be kept at reset value.

Bits 20:16 **CSBOUND[4:0]:** NCS boundary
This bitfield enables the transaction boundary feature. When active, a minimum value of 3 is recommended. The NCS is released on each boundary of $2^{CSBOUND}$ bytes.
0: NCS boundary disabled
Others: NCS boundary set to $2^{CSBOUND}$ bytes

Bits 15:0: Reserved, must be kept at reset value.

23.7.5 OCTOSPI device configuration register 4 (OCTOSPI_DCR4)

Address offset: 0x0014
Reset value: 0x0000 0000
This register can be modified only when BUSY = 0.

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

| REFRESH[15:0] | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

---

Bits 7:0 **PRESCALER[7:0]:** Clock prescaler
This bitfield defines the scaler factor for generating the CLK based on the kernel clock (value + 1).
0: $F_{CLK} = F_{KERNEL}$, kernel clock used directly as OCTOSPI CLK (prescaler bypassed). In this case, if the DTR mode is used, it is mandatory to provide to the OCTOSPI a kernel clock that has 50% duty-cycle.
1: $F_{CLK} = F_{KERNEL}/2$
2: $F_{CLK} = F_{KERNEL}/3$
...
255: $F_{CLK} = F_{KERNEL}/256$
For odd clock division factors, the CLK duty cycle is not 50%. The clock signal remains low one cycle longer than it stays high.

---

**RM0481**

Octo-SPI interface (OCTOSPI)
Bits 31:0  REFRESH[31:0]: Refresh rate

This bitfield enables the refresh rate feature. The NCS is released every REFRESH + 1 clock cycles for writes, and REFRESH + 4 clock cycles for reads. These two values can be extended with few clock cycles when refresh occurs during a byte transmission in single-, dual- or quad-SPI mode, because the byte transmission must be completed.

0: Refresh disabled
Others: Maximum communication length is set to REFRESH + 1 clock cycles.

Note: REFRESH count is based on the divided clock period: if OCTOSPI_DCR2 PRESCALER bitfield is changed, the REFRESH field must be updated accordingly.

### 23.7.6 OCTOSPI status register (OCTOSPI_SR)

Address offset: 0x0020

Reset value: 0x0000 0000

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

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8  FLEVEL[5:0]: FIFO level

This bitfield gives the number of valid bytes that are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 32 when it is full.

In automatic status-polling mode, FLEVEL is zero.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5  BUSY: Busy

This bit is set when an operation is ongoing. It is cleared automatically when the operation with the external device is finished and the FIFO is empty.

Bit 4  TOF: Timeout flag

This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3  SMF: Status match flag

This bit is set in automatic status-polling mode when the unmasked received data matches the corresponding bits in the match register (OCTOSPI_PSMAR).

It is cleared by writing 1 to CSMF.

Bit 2  FTF: FIFO threshold flag

In indirect mode, this bit is set when the FIFO threshold has been reached, or if there is any data left in the FIFO after the reads from the external device are complete.

It is cleared automatically as soon as the threshold condition is no longer true.

In automatic status-polling mode, this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1  TCF: Transfer complete flag

This bit is set in indirect mode when the programmed number of data has been transferred or in any mode when the transfer has been aborted. It is cleared by writing 1 to CTCF.
Bit 0 **TEF**: Transfer error flag
This bit is set in indirect mode when an invalid address is being accessed in indirect mode.
It is cleared by writing 1 to CTEF.

### 23.7.7 OCTOSPI flag clear register (OCTOSPI_FCR)
Address offset: 0x0024
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15</td>
<td><strong>CTOF</strong>: Clear timeout flag</td>
</tr>
<tr>
<td>14</td>
<td>Writing 1 clears the TOF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>13</td>
<td><strong>CSMF</strong>: Clear status match flag</td>
</tr>
<tr>
<td>12</td>
<td>Writing 1 clears the SMF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>11</td>
<td><strong>CTCF</strong>: Clear transfer complete flag</td>
</tr>
<tr>
<td>10</td>
<td>Writing 1 clears the TCF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>9</td>
<td><strong>CTEF</strong>: Clear transfer error flag</td>
</tr>
<tr>
<td>8</td>
<td>Writing 1 clears the TEF flag in the OCTOSPI_SR register.</td>
</tr>
</tbody>
</table>

### 23.7.8 OCTOSPI data length register (OCTOSPI_DLR)
Address offset: 0x0040
Reset value: 0x0000 0000
This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td><strong>DL[31:16]</strong>: Data length</td>
</tr>
<tr>
<td>30</td>
<td>Writing 1 clears the TOF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>29</td>
<td>Writing 1 clears the SMF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>28</td>
<td>Writing 1 clears the TCF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>27</td>
<td>Writing 1 clears the TEF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>26</td>
<td>Reading 1 clears the TOF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>25</td>
<td>Reading 1 clears the SMF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>24</td>
<td>Reading 1 clears the TCF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>23</td>
<td>Reading 1 clears the TEF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>22</td>
<td>Reading 1 clears the TOF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>21</td>
<td>Reading 1 clears the SMF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>20</td>
<td>Reading 1 clears the TCF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>19</td>
<td>Reading 1 clears the TEF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>18</td>
<td>Reading 1 clears the TOF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>17</td>
<td>Reading 1 clears the SMF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>16</td>
<td>Reading 1 clears the TCF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>15</td>
<td>Reading 1 clears the TEF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>14</td>
<td>Reading 1 clears the TOF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>13</td>
<td>Reading 1 clears the SMF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>12</td>
<td>Reading 1 clears the TCF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>11</td>
<td>Reading 1 clears the TEF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>10</td>
<td>Writing 1 clears the TOF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>9</td>
<td>Writing 1 clears the SMF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>8</td>
<td>Writing 1 clears the TCF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>7</td>
<td>Writing 1 clears the TEF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>6</td>
<td>Writing 1 clears the TOF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>5</td>
<td>Writing 1 clears the SMF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>4</td>
<td>Writing 1 clears the TCF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>3</td>
<td>Writing 1 clears the TEF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>2</td>
<td>Writing 1 clears the TOF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>1</td>
<td>Writing 1 clears the SMF flag in the OCTOSPI_SR register.</td>
</tr>
<tr>
<td>0</td>
<td>Writing 1 clears the TCF flag in the OCTOSPI_SR register.</td>
</tr>
</tbody>
</table>
23.7.9 OCTOSPI address register (OCTOSPI_AR)

Address offset: 0x0048  
Reset value: 0x0000 0000  
This register can be modified only when BUSY = 0 and FMODE ≠ 11.

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

Bits 31:0 ADDRESS[31:0]: Address  
Address to be sent to the external device. In HyperBus protocol, this field must be even as this protocol is 16-bit word oriented. In dual-memory configuration, AR[0] is forced to 0.

23.7.10 OCTOSPI data register (OCTOSPI_DR)

Address offset: 0x0050  
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>DATA[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 31:0 DL[31:0]: Data length  
Number of data to be retrieved (value+1) in indirect and automatic status-polling modes. A value not greater than three (indicating 4 bytes) must be used for automatic status-polling mode.  
All 1’s in indirect mode means undefined length, where OCTOSPI continues until the end of the memory, as defined by DEVSIZE.  
0x0000_0000: 1 byte is to be transferred.  
0x0000_0001: 2 bytes are to be transferred.  
0x0000_0002: 3 bytes are to be transferred.  
0x0000_0003: 4 bytes are to be transferred.  
...  
0xFFFF_FFD: 4,294,967,294 (4G-2) bytes are to be transferred.  
0xFFFF_FFE: 4,294,967,295 (4G-1) bytes are to be transferred.  
0xFFFF_FFFF: undefined length; all bytes, until the end of the external device, (as defined by DEVSIZE) are to be transferred. Continue reading indefinitely if DEVSIZE = 0x1F.  
DL[0] is stuck at 1 in dual-memory configuration (DMM = 1) even when 0 is written to this bit, thus assuring that each access transfers an even number of bytes.  
This bitfield has no effect in memory-mapped mode.
23.7.11 OCTOSPI polling status mask register (OCTOSPI_PSMKR)

Address offset: 0x0080
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bits 31:0 MASK[31:0]: Status mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mask to be applied to the status bytes received in automatic status-polling mode</td>
</tr>
<tr>
<td>For bit n:</td>
</tr>
<tr>
<td>0: Bit n of the data received in automatic status-polling mode is masked and its value is not considered in the matching logic.</td>
</tr>
<tr>
<td>1: Bit n of the data received in automatic status-polling mode is unmasked and its value is considered in the matching logic.</td>
</tr>
</tbody>
</table>
23.7.12 OCTOSPI polling status match register (OCTOSPI_PSMAR)

Address offset: 0x0088
Reset value: 0x0000 0000
This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bits 31:0 MATCH[31:16]</th>
<th>Bits 15:0 MATCH[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw</td>
<td>rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8</td>
<td>7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:0 MATCH[31:0]: Status match
Value to be compared with the masked status register to get a match

23.7.13 OCTOSPI polling interval register (OCTOSPI_PIR)

Address offset: 0x0090
Reset value: 0x0000 0000
This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bits 31:16 Reserved</th>
<th>Bits 15:0 INTERVAL[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw</td>
<td>rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9</td>
<td>8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.
Bits 15:0 INTERVAL[15:0]: Polling interval
Number of CLK cycles between a read during the automatic status-polling phases

23.7.14 OCTOSPI communication configuration register (OCTOSPI_CCR)

Address offset: 0x0100
Reset value: 0x0000 0000
This register can be modified only when BUSY = 0.
Bits 31:30  Reserved, must be kept at reset value.

Bit 29  **DQSE**: DQS enable
This bit enables the data strobe management.
- 0: DQS disabled
- 1: DQS enabled

Bit 28  Reserved, must be kept at reset value.

Bit 27  **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
- 0: DTR mode disabled for data phase
- 1: DTR mode enabled for data phase

Bits 26:24  **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
- 000: No data
- 001: Data on a single line
- 010: Data on two lines
- 011: Data on four lines
- 100: Data on eight lines
- Others: Reserved

Bits 23:22  Reserved, must be kept at reset value.

Bits 21:20  **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
- 00: 8-bit alternate bytes
- 01: 16-bit alternate bytes
- 10: 24-bit alternate bytes
- 11: 32-bit alternate bytes

Bit 19  **ABDTR**: Alternate-byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
- 0: DTR mode disabled for the alternate-byte phase
- 1: DTR mode enabled for the alternate-byte phase

Bits 18:16  **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
- 000: No alternate bytes
- 001: Alternate bytes on a single line
- 010: Alternate bytes on two lines
- 011: Alternate bytes on four lines
- 100: Alternate bytes on eight lines
- Others: Reserved

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

Bits 13:12  **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
- 00: 8-bit address
- 01: 16-bit address
- 10: 24-bit address
- 11: 32-bit address
23.7.15 OCTOSPI timing configuration register (OCTOSPI_TCR)

Address offset: 0x0108
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

Bit 31 Reserved, must be kept at reset value.
Bit 30 **SSSHIFT**: Sample shift

By default, the OCTOSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The software must ensure that SSHIFT = 0 when the data phase is configured in DTR mode (when DDTR = 1.)

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle
0: No delay hold
1: 1/4 cycle hold

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

### 23.7.16 OCTOSPI instruction register (OCTOSPI_IR)

Address offset: 0x0110
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

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

Bits 31:0 **INSTRUCTION[31:0]**: Instruction

Instruction to be sent to the external SPI device

### 23.7.17 OCTOSPI alternate bytes register (OCTOSPI_ABR)

Address offset: 0x0120
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

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

**INSTRUCTION**

```plaintext
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw

15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw
```
23.7.18 OCTOSPI low-power timeout register (OCTOSPI_LPTR)

Address offset: 0x00130
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>ALTERNATE[31:0]</td>
<td>Alternate bytes: Optional data to be sent to the external SPI device right after the address.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>TIMEOUT[15:0]</td>
<td>Timeout period: After each access in memory-mapped mode, the OCTOSPI prefetches the subsequent bytes and hold them in the FIFO. This bitfield indicates how many CLK cycles the OCTOSPI waits after the clock becomes inactive and until it raises the NCS, putting the external device in a lower-consumption state.</td>
</tr>
</tbody>
</table>

23.7.19 OCTOSPI wrap communication configuration register (OCTOSPI_WPCCR)

Address offset: 0x0140
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 29</td>
<td>DQSE</td>
<td>DQS enable: This bit enables the data strobe management.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: DQS disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: DQS enabled</td>
</tr>
<tr>
<td>Bit 28</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bit 27  **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase

Bits 26:24  **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Others: Reserved

Bits 23:22  Reserved, must be kept at reset value.

Bits 21:20  **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19  **ABDTR**: Alternate-byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16  **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: no alternate bytes
001: alternate bytes on a single line
010: alternate bytes on two lines
011: alternate bytes on four lines
100: alternate bytes on eight lines
Others: reserved

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

Bits 13:12  **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11  **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for address phase
1: DTR mode enabled for address phase
Bits 10:8 **ADMODE[2:0]**: Address mode

This bitfield defines the address phase mode of operation.

- **000**: No address
- **001**: Address on a single line
- **010**: Address on two lines
- **011**: Address on four lines
- **100**: Address on eight lines
- Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size

This bitfield defines the instruction size.

- **00**: 8-bit instruction
- **01**: 16-bit instruction
- **10**: 24-bit instruction
- **11**: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate

This bit sets the DTR mode for the instruction phase.

- **0**: DTR mode disabled for the instruction phase
- **1**: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode

This bitfield defines the instruction phase mode of operation.

- **000**: No instruction
- **001**: Instruction on a single line
- **010**: Instruction on two lines
- **011**: Instruction on four lines
- **100**: Instruction on eight lines
- Others: Reserved

### 23.7.20 OCTOSPI wrap timing configuration register (OCTOSPI_WPTCR)

Address offset: 0x0148

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

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

Bit 31 Reserved, must be kept at reset value.
Bit 30  **SSHIFT**: Sample shift
By default, the OCTOSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The firmware must assure that SSHIFT=0 when the data phase is configured in DTR mode (when DDTR = 1).

Bit 29  Reserved, must be kept at reset value.

Bit 28  **DHQC**: Delay hold quarter cycle
Add a quarter cycle delay on the outputs in DTR communication to match hold requirement.
0: No quarter cycle delay
1: 1/4 cycle delay inserted

Bits 27:5  Reserved, must be kept at reset value.

Bits 4:0  **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

### 23.7.21 OCTOSPI wrap instruction register (OCTOSPI_WPIR)

Address offset: 0x0150
Reset value: 0x0000 0000
This register can be modified only when BUSY = 0.

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

### 23.7.22 OCTOSPI wrap alternate bytes register (OCTOSPI_WPABR)

Address offset: 0x0160
Reset value: 0x0000 0000
This register can be modified only when BUSY = 0.

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

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
**OCTOSPI write communication configuration register (OCTOSPI_WCCR)**

Address offset: 0x0180  
Reset value: 0x0000 0000  

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

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

**Bits 31:0**  
**ALTERNATE[31:0]: Alternate bytes**  
Optional data to be sent to the external SPI device right after the address

---

**Bit 29**  
**DQSE** (DQS enable)  
This bit enables the data strobe management.  
0: DQS disabled  
1: DQS enabled

**Bit 28**  
Reserved, must be kept at reset value.

**Bit 27**  
**DDTR** (data double transfer rate)  
This bit sets the DTR mode for the data phase.  
0: DTR mode disabled for the data phase  
1: DTR mode enabled for the data phase

**Bits 26:24**  
**DMODE[2:0]: Data mode**  
This bitfield defines the data phase mode of operation.  
000: No data  
001: Data on a single line  
010: Data on two lines  
011: Data on four lines  
100: Data on eight lines  
Others: Reserved

**Bits 23:22**  
Reserved, must be kept at reset value.

**Bits 21:20**  
**ABSIZE[1:0]: Alternate-byte size**  
This bitfield defines the alternate-byte size.  
00: 8-bit alternate bytes  
01: 16-bit alternate bytes  
10: 24-bit alternate bytes  
11: 32-bit alternate bytes

---

**Bits 31:30**  
Reserved, must be kept at reset value.

---

**Bit 29**  
**DQSE**: DQS enable  
This bit enables the data strobe management.  
0: DQS disabled  
1: DQS enabled

**Bit 28**  
Reserved, must be kept at reset value.

**Bit 27**  
**DDTR**: data double transfer rate  
This bit sets the DTR mode for the data phase.  
0: DTR mode disabled for the data phase  
1: DTR mode enabled for the data phase

**Bits 26:24**  
**DMODE[2:0]: Data mode**  
This bitfield defines the data phase mode of operation.  
000: No data  
001: Data on a single line  
010: Data on two lines  
011: Data on four lines  
100: Data on eight lines  
Others: Reserved

**Bits 23:22**  
Reserved, must be kept at reset value.

**Bits 21:20**  
**ABSIZE[1:0]: Alternate-byte size**  
This bitfield defines the alternate-byte size.  
00: 8-bit alternate bytes  
01: 16-bit alternate bytes  
10: 24-bit alternate bytes  
11: 32-bit alternate bytes
Bit 19 **ABDTR**: Alternate bytes double transfer rate
This bit sets the DTR mode for the alternate-bytes phase.
0: DTR mode disabled for alternate-bytes phase
1: DTR mode enabled for alternate-bytes phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Others: Reserved

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

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for instruction phase
1: DTR mode enabled for instruction phase
23.7.24 OCTOSPI write timing configuration register (OCTOSPI_WTCR)

Address offset: 0x0188
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>INSTRUCTION[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>DCYC[4:0]</td>
</tr>
<tr>
<td></td>
<td>rw rw rw rw rw</td>
</tr>
</tbody>
</table>

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

Bits 4:0 DCYC[4:0]: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency. In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

23.7.25 OCTOSPI write instruction register (OCTOSPI_WIR)

Address offset: 0x0190
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>INSTRUCTION[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>INSTRUCTION[15:0]</td>
</tr>
<tr>
<td></td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 31:0 INSTRUCTION[31:0]: Instruction
Instruction to be sent to the external SPI device
23.7.26 OCTOSPI write alternate bytes register (OCTOSPI_WABR)

Address offset: 0x01A0
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

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

Bits 31:0 ALTERNATE[31:0]: Alternate bytes
Optional data to be sent to the external SPI device right after the address

23.7.27 OCTOSPI HyperBus latency configuration register (OCTOSPI_HLCR)

Address offset: 0x0200
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 TRWR[7:0]: Read-write minimum recovery time
Device read-to-write/write-to-read minimum recovery time expressed in number of communication clock cycles

Bits 15:8 TACC[7:0]: Access time
Device access time according to the memory latency, expressed in number of communication clock cycles

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

Bit 1 WZL: Write zero latency
This bit enables zero latency on write operations.
0: Latency on write accesses
1: No latency on write accesses
**Bit 0 LM: Latency mode**
This bit selects the latency mode.
0: Variable initial latency
1: Fixed latency

### 23.7.28 OCTOSPI register map

Table 209. OCTOSPI register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Format</th>
<th>Offset</th>
<th>Name</th>
<th>Bit 0</th>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
<th>Bit 4</th>
<th>Bit 5</th>
<th>Bit 6</th>
<th>Bit 7</th>
<th>Bit 8</th>
<th>Bit 9</th>
<th>Bit 10</th>
<th>Bit 11</th>
<th>Bit 12</th>
<th>Bit 13</th>
<th>Bit 14</th>
<th>Bit 15</th>
<th>Bit 16</th>
<th>Bit 17</th>
<th>Bit 18</th>
<th>Bit 19</th>
<th>Bit 20</th>
<th>Bit 21</th>
<th>Bit 22</th>
<th>Bit 23</th>
<th>Bit 24</th>
<th>Bit 25</th>
<th>Bit 26</th>
<th>Bit 27</th>
<th>Bit 28</th>
<th>Bit 29</th>
<th>Bit 30</th>
<th>Bit 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>OCTOSPI_CR</td>
<td></td>
<td>31</td>
<td>name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset</td>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0004</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0008</td>
<td>OCTOSPI_DCR1</td>
<td></td>
<td>31</td>
<td>name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset</td>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x000C</td>
<td>OCTOSPI_DCR2</td>
<td></td>
<td></td>
<td>31</td>
<td>name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset</td>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0010</td>
<td>OCTOSPI_DCR3</td>
<td></td>
<td>31</td>
<td>name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset</td>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0014</td>
<td>OCTOSPI_DCR4</td>
<td></td>
<td>31</td>
<td>name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset</td>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0018-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x001C</td>
<td>OCTOSPI_SR</td>
<td></td>
<td>31</td>
<td>name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset</td>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0024</td>
<td>OCTOSPI_FCR</td>
<td></td>
<td>31</td>
<td>name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset</td>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0028-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x003C</td>
<td>OCTOSPI_DLR</td>
<td></td>
<td>31</td>
<td>name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset</td>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0044</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0048</td>
<td>OCTOSPI_AR</td>
<td></td>
<td>31</td>
<td>name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset</td>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x004C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>Offset Register</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>----------------</td>
<td>-------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0050</td>
<td>OCTOSPI_DR</td>
<td>DATA[31:0]</td>
<td>Reset value: 00000000000000000000000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0054-0x007C</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0080</td>
<td>OCTOSPI_PSMKR</td>
<td>MASK[31:0]</td>
<td>Reset value: 00000000000000000000000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0084</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0088</td>
<td>OCTOSPI_PSMAR</td>
<td>MATCH[31:0]</td>
<td>Reset value: 00000000000000000000000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008C</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0090</td>
<td>OCTOSPI_PIR</td>
<td>INTERVAL[15:0]</td>
<td>Reset value: 0000000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0094-0x00FC</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0100</td>
<td>OCTOSPI_CCR</td>
<td>DQSE DDTR DMODE [2:0]</td>
<td>Reset value: 0000000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0104</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0108</td>
<td>OCTOSPI_TCR</td>
<td>SSHIFT DHQC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010C</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0110</td>
<td>OCTOSPI_IR</td>
<td>INSTRUCTION[31:0]</td>
<td>Reset value: 00000000000000000000000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0114-0x011C</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0120</td>
<td>OCTOSPI_ABR</td>
<td>ALTERNATE[31:0]</td>
<td>Reset value: 00000000000000000000000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0124-0x012C</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0130</td>
<td>OCTOSPI_LPTR</td>
<td>TIMEOUT[15:0]</td>
<td>Reset value: 0000000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0134-0x013C</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0140</td>
<td>OCTOSPI_WPCCR</td>
<td>DQSE DDTR DMODE [2:0]</td>
<td>Reset value: 0000000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0144</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Refer to Section 2.3 for the register boundary addresses.
24 Secure digital input/output MultiMediaCard interface (SDMMC)

24.1 SDMMC main features

The SD/SDIO, embedded MultiMediaCard (eMMC) host interface (SDMMC) provides an interface between the AHB bus and SD memory cards, SDIO cards and eMMC devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association website at www.jedec.org, published by the MMCA technical committee.

SD memory card and SD I/O card system specifications are available through the SD card Association website at www.sdcard.org.

The SDMMC features include the following:

- Compliance with *Embedded MultiMediaCard System Specification Version 5.1*. Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit. (HS200 SDMMC_CK speed limited to maximum allowed I/O speed) (HS400 is not supported).

- Full compatibility with previous versions of MultiMediaCards (backward compatibility).

- Full compliance with *SD memory card specifications version 6.0*. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).

- Full compliance with *SDIO card specification version 4.0*. Card support for two different databus modes: 1-bit (default) and 4-bit. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).

- Data transfer up to 208 Mbyte/s for the 8-bit mode. (depending maximum allowed I/O speed).

- Data and command output enable signals to control external bidirectional drivers.

- IDMA linked list support

The MultiMediaCard/SD bus connects cards to the host.

The current version of the SDMMC supports only one SD/SDIO/eMMC card at any one time and a stack of eMMC.

24.2 SDMMC implementation

<table>
<thead>
<tr>
<th>SDMMC modes/features(1)</th>
<th>SDMMC1</th>
<th>SDMMC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable delay (SRD104, HS200)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SDMMC_CKIN</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SDMMC_CDIR, SDMMC_D0DIR</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>SDMMC_D123DIR</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

1. X = supported.
24.3 SDMMC bus topology

Communication over the bus is based on command/response and data transfers.

The basic transaction on the SD/SDIO/eMMC bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers are done in the following ways:
- Block mode: data block(s) with block size $2^N$ bytes with $N$ in the range 0-14.
- SDIO multibyte mode: single data block with block size range 1-512 bytes
- eMMC Stream mode: continuous data stream

Data transfers to/from eMMC cards are done in data blocks or streams.

Figure 152. SDMMC “no response” and “no data” operations

Figure 153. SDMMC (multiple) block read operation

Note: The Stop Transmission command is not required at the end of an eMMC multiple block read with predefined block count.
Figure 154. SDMMC (multiple) block write operation

Note: The Stop Transmission command is not required at the end of an eMMC multiple block write with predefined block count.

Note: The SDMMC does not send any data as long as the Busy signal is asserted (SDMMC_D0 pulled low).

Figure 155. SDMMC (sequential) stream read operation

Stream data transfer operates only in a 1-bit wide bit bus configuration on SDMMC_D0 in single data rate modes (DS, HS, and SDR).
# SDMMC operation modes

## Table 211. SDMMC operation modes SD and SDIO

<table>
<thead>
<tr>
<th>SDIO Bus Speed modes (1)(2)</th>
<th>Max Bus Speed (3) [Mbyte/s]</th>
<th>Max Clock frequency (4) [MHz]</th>
<th>Signal Voltage [V]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS (Default Speed)</td>
<td>12.5</td>
<td>25</td>
<td>3.3</td>
</tr>
<tr>
<td>HS (High Speed)</td>
<td>25</td>
<td>50</td>
<td>3.3</td>
</tr>
<tr>
<td>SDR12</td>
<td>12.5</td>
<td>25</td>
<td>1.8</td>
</tr>
<tr>
<td>SDR25</td>
<td>25</td>
<td>50</td>
<td>1.8</td>
</tr>
<tr>
<td>DDR50</td>
<td>50</td>
<td>50</td>
<td>1.8</td>
</tr>
<tr>
<td>SDR50</td>
<td>50</td>
<td>100</td>
<td>1.8</td>
</tr>
<tr>
<td>SDR104</td>
<td>104</td>
<td>208</td>
<td>1.8</td>
</tr>
</tbody>
</table>

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC_CK clock edges).
3. SDIO bus speed with 4bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.

SDR104 mode requires variable delay support using sampling point tuning. The use of variable delay is optional for SDR50 mode.

## Table 212. SDMMC operation modes eMMC

<table>
<thead>
<tr>
<th>eMMC bus speed modes (1)(2)</th>
<th>Max bus speed (3) [Mbyte/s]</th>
<th>Max clock frequency (4) [MHz]</th>
<th>Signal voltage (5) [V]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legacy compatible</td>
<td>26</td>
<td>26</td>
<td>3/1.8/1.2V</td>
</tr>
<tr>
<td>High speed SDR</td>
<td>52</td>
<td>52</td>
<td>3/1.8/1.2V</td>
</tr>
<tr>
<td>High speed DDR</td>
<td>104</td>
<td>52</td>
<td>3/1.8/1.2V</td>
</tr>
<tr>
<td>High speed HS200</td>
<td>200</td>
<td>200</td>
<td>1.8/1.2V</td>
</tr>
</tbody>
</table>

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC_CK clock edges).
3. eMMC bus speed with 8bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.
5. Supported signal voltage level depends on I/O port characteristics, refer to device datasheet.

HS200 mode requires variable delay support using sampling point tuning.
24.5 SDMMC functional description

The SDMMC consists of four parts:

- The AHB slave interface accesses the SDMMC adapter registers, and generates interrupt signals and IDMA control signals.
- The SDMMC adapter block provides all functions specific to the eMMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
- The internal DMA (IDMA) block with its AHB master interface.
- A delay block (DLYB) taking care of the receive data sample clock alignment. The delay block is NOT part of the SDMMC. A delay block is mandatory when supporting SDR104 or HS200.

24.5.1 SDMMC block diagram

Figure 157 shows the SDMMC block diagram.

![SDMMC block diagram](image)

24.5.2 SDMMC pins and internal signals

Table 213 lists the SDMMC internal input/output signals, Table 214 the SDMMC pins (alternate functions).

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sdmmc_ker_ck</td>
<td>Digital input</td>
<td>SDMMC kernel clock</td>
</tr>
<tr>
<td>sdmmc_hclk</td>
<td>Digital input</td>
<td>AHB clock</td>
</tr>
<tr>
<td>sdmmc_it</td>
<td>Digital output</td>
<td>SDMMC global interrupt</td>
</tr>
</tbody>
</table>
24.5.3 General description

The SDMMC lines have different operating modes:

- By default, SDMMC_D0 line is used for data transfer. After initialization, the host can change the data bus width.
- For an eMMC, 1-bit (SDMMC_D0), 4-bit (SDMMC_D[3:0]) or 8-bit (SDMMC_D[7:0]) data bus widths can be used.
- For an SD or an SDIO card, 1-bit (SDMMC_D0) or 4-bit (SDMMC_D[3:0]) can be used. All data lines operate in push-pull mode.

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the data lines is indicated with I/O direction signals. The SDMMC_D0DIR signal indicates the I/O direction for the SDMMC_D0 data line, the SDMMC_D123DIR for the SDMMC_D[3:1] data lines.

SDMMC_CMD only operates in push-pull mode:

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the SDMMC_CMD line is indicated with the I/O direction signal SDMMC_CDIR.
SDMMC\_CK clock to the card originates from `sdmmc\_ker\_ck`:
- When the `sdmmc\_ker\_ck` clock has 50 % duty cycle, it can be used even in bypass mode (CLKDIV = 0).
- When the `sdmmc\_ker\_ck` duty cycle is not 50 %, the CLKDIV must be used to divide it by 2 or more (CLKDIV > 0).
- The phase relation between the SDMMC\_CMD / SDMMC\_D[7:0] outputs and the SDMMC\_CK can be selected through the NEGEDGE bit. The phase relation depends on the CLKDIV, NEGEDGE, and DDR settings. See Figure 158.

**Figure 158. SDMMC Command and data phase relation**

![Figure 158](image)

**Table 215. SDMMC Command and data phase selection**

<table>
<thead>
<tr>
<th>CLKDIV</th>
<th>DDR</th>
<th>NEGEDGE</th>
<th>Command out</th>
<th>Data out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 x x  =</td>
<td>sdmmc_ker_ck</td>
<td>generated on sdmmc_ker_ck falling edge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;0 0 0</td>
<td>generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;0 1 1</td>
<td>generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;0 0 1</td>
<td>generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.</td>
<td>generated on sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;0 1 1</td>
<td>generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

By default, the `sdmmc\_io\_in\_ck` feedback clock input is selected for sampling incoming data in the SDMMC receive path. It is derived from the SDMMC\_CK pin.

For tuning the phase of the sampling clock to accommodate the receive data timing, the DLYB delay block available on the device can be connected between `sdmmc\_io\_in\_ck` signal (DLYB input dlyb\_in\_ck) and `sdmmc\_fb\_ck` clock input of SDMMC (DLYB output dlyb\_out\_ck). Selecting the `sdmmc\_fb\_ck` clock input in the receive path then enables using the phase-tuned sampling clock for the incoming data. This is required for SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.
When using an external driver (a voltage switch transceiver), the SDMMC_CKIN feedback clock input can be selected to sample the receive data.

For an SD/SDIO/eMMC card, the clock frequency can vary between 0 and 208 MHz (limited by maximum I/O speed).

Depending on the selected bus mode (SDR or DDR), one bit or two bits are transferred on SDMMC_D[7:0] lines with each clock cycle. The SDMMC_CMD line transfers only one bit per clock cycle.

24.5.4 SDMMC adapter

The SDMMC adapter (see Figure 157: SDMMC block diagram) is a multimedia/secure digital memory card bus master that provides an interface to a MultiMediaCard stack or to a secure digital memory card. It consists of the following subunits:

- Control unit
- Data transmit path
- Command path
- Data receive path
- Response path
- Receive data path clock multiplexer
- Delay block (DLYB), external to the SDMMC
- Adapter register block
- Data FIFO
- Internal DMA (IDMA)

Note: The adapter registers and FIFO use the AHB clock domain (sdmmc_hclk). The control unit, command path and data transmit path use the SDMMC adapter clock domain (sdmmc_ker_ck). The response path and data receive path use the SDMMC adapter feedback clock domain from the sdmmc_io_in_ck, or SDMMC_CKIN, or from the sdmmc_fb_ck generated by DLYB.

The DLYB delay block on the device can be used in conjunction with the SDMMC adapter, to tune the phase of the sampling clock for incoming data in SDMMC receive mode. It is required for the SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

Adapter register block

The adapter register block contains all system control registers, the SDMMC command and response registers and the data FIFO.

This block also generates the signals from the corresponding bit location in the SDMMC Clear register that clear the static flags in the SDMMC adapter.

Control unit

The control unit illustrated in Figure 159, contains the power management functions, the SDMMC_CK clock management with divider, and the I/O direction management.
The power management subunit disables the card bus output signals during the power-off and power-up phases.

There are three power phases:
- power-off
- power-up
- power-on

The clock management subunit uses the sdmmc_ker_ck to generate the SDMMC_CK and provides the division control. It also takes care of stopping the SDMMC_CK for i.e. flow control.

The clock outputs are inactive:
- after reset
- during the power-off or power-up phases
- if the power saving mode (register bit PWRSAV) is enabled and the card bus is in the Idle state for eight clock periods. The clock is stopped eight cycles after both the command/response CPSM and data path DPSM subunits have enter the Idle phase. The clock is restarted when the command/response CPSM or data path DPSM is activated (enabled).

The I/O management subunit takes care of the SDMMC_Dn and SDMMC_CMD I/O direction signals, which controls the external voltage transceiver.

**Command/response path**

The command/response path subunit transfers commands and responses on the SDMMC_CMD line. The command path is clocked on the SDMMC_CK and sends commands to the card. The response path is clocked on the sdmmc_rx_ck and receives responses from the card.
Command/response path state machine (CPSM)

- When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent the CRC is appended and the command path state machine (CPSM) sets the status flags and:
  - if a response is not required enters the Idle state.
  - If a response is required, it waits for the response.

- When the response is received,
  - for a response with CRC, the received CRC code and the internally generated code are compared, and the appropriate status flag is set according the result.
  - for a response without CRC, no CRC is checked, and the appropriate status flag is not set.

When ever the CPSM is active, i.e. not in the Idle state, the CPSMACT bit is set.
• **Idle**: The command path is inactive. When the command control register is written and the enable bit (CPSMEN) is set, the CPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit) and moves
  – to the Send state when WAITPEND = 0 and BOOTEN = 0.
  – to the Pending state when WAITPEND = 1.
  – to the Boot state when BOOTEN = 1.

• **Send**: The command is sent and the CRC is appended.
  – When CMDTRANS bit is set or when BOOTEN bit is set and BOOTMODE is alternative boot, and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM at the end of the command.
  – When the CMDTRANS bit is set and the CMD_SUSPEND bit is 0 the interrupt period is terminated at the end of the command.
  – When CMDSTOP bit is set the CPSM Abort signal is issued to the DPSM at the end of the command.
  – If no response is expected (WAITRESP = 00) the CPSM moves to the Idle state and the CMDSENT flag is set. When BOOTMODE = 1 and BOOTEN = 0 the CMDSENT flag is delayed 56 cycles after the command end bit, otherwise the
CMDSENT flag is generated immediately after the command end bit. The RESPCMDR and RESPxR registers are not modified.
- If a command response is expected (WAITRESP = not 00) the CPSM moves to the Wait state and start the response timeout.

- **Wait:** The command path waits for a response.
  - When WAITINT bit is 0 the command timer starts running and the CPSM waits for a start bit.
    a) If a start bit is detected before the timeout the CPSM moves to the Receive state.
    b) If the timeout is reached before the CPSM detect a response start bit, the timeout flag (CTIMEOUT) is set and the CPSM moves to the Idle state.
    The RESPCMDR and RESPxR registers are not modified.
  - When WAITINT bit is 1, the timer is disabled and the CPSM waits for an interrupt request (response start bit) from one of the cards.
    a) When a start bit is detected the CPSM moves to the Receive state.
    b) When writing WAITINT to 0 (interrupt mode abort), the host sends a response by its self and on detecting the start bit the CPSM move to the Receive state.

- **Receive:** The command response is received. Depending the response mode bits WAITRESP in the command control register, the response can be either short or long, with CRC or without CRC. The received CRC code when present is verified against the internally generated CRC code.
  - When the CMDGSUSPEND bit is set and the SDIO Response bit BS = 0 (response bit [39]), the interrupt period is started after the response.
    When the CMDGSUSPEND bit is cleared, or the CMDGSUSPEND bit is 1 and the SDIO Response bit BS = 1 (response bit [39]), there is no interrupt period started.
  - When the CMDTRANS bit is set and the CMDGSUSPEND bit is set and the SDIO Response bit DF= 1 (response bit [32]) the interrupt period is terminated after the response.
  - When the CRC status passes or no CRC is present the CMDREND flag is set, the CPSM moves to the Idle state.
    The RESPCMDR and RESPxR registers are updated with received response.
    - When BOOTMODE = 1 and BOOTEN = 0 the CMDREND flag is delayed 56 cycles after the response end bit, otherwise the CMDREND flag is generated immediately after the response end bit.
    - When CMDTRANS bit is set and the DTDIR = transmit, the CPSM DataEnable signal is issued to the DPSM at the end of the command response.
  - When the CRC status fails the CCRCFAIL flag is set and the CPSM moves to the Idle state.
    The RESPCMDR and RESPxR registers are updated with received response.

- **Pending:** According the pending WAITPEND bit in the command register, the CPSM enters the pending state.
  - When DATALENGTH <= 5 bytes the CPSM moves to the Sent state and generates the DataEnable signal to start the data transfer aligned with the CMD12 Stop Transmission command.
  - When DATALENGTH > 5 bytes, the CPSM DataEnable signal is issued to the DPSM to start the data transfer. The CPSM waits for a send CMD signal from the
DPSM before moving to the Send state. This enables i.e. the CMD12 Stop Transmission command to be sent aligned with the data.

– When writing WAITPEND to 0, the CPSM moves to the Send state.

• **Boot:** If the BOOTEN bit is set in the command register, the CPSM enters the Boot state, and when:
  – BOOTMODE = 0 the SDMMC_CMD line is driven low and when CMDTRANS bit is set and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM. This enables normal boot operation. This state is left at the end of the boot procedure by clearing the register bit BOOTEN, which cause the SDMMC_CMD line to be driven high and the CPSM Abort signal is issued to the DPSM, before moving to the Idle state. The CMDSENT flag is generated 56 cycles after SDMMC_CMD line is high.
  – BOOTMODE = 1, move to the Send state. This enables sending of the CMD0 (boot). Clearing BOOTEN has no effect.

**Note:** The CPSM remains in the Idle state for at least eight SDMMC_CK periods to meet the NCC and NRC timing constraints. NCC is the minimum delay between two host commands, and NRC is the minimum delay between the host command and the card response.

**Note:** The response timeout has a fixed value of 64 SDMMC_CK clock periods.

A command is a token that starts an operation. Commands are sent from the host to either a single card (addressed command) or all connected cards (broadcast command are available for e•MMC V3.31 or previous). Commands are transferred serially on the SDMMC_CMD line. All commands have a fixed length of 48 bits. The general format for a command token for SD-Memory cards, SDIO cards, and e•MMC cards is shown in Table 216.

The command token data is taken from 2 registers, one containing a 32-bits argument and the other containing the 6-bits command index (six bits sent to a card).

### Table 216. Command token format

<table>
<thead>
<tr>
<th>Bit position</th>
<th>Width</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>1</td>
<td>0</td>
<td>Start bit</td>
</tr>
<tr>
<td>46</td>
<td>1</td>
<td>1</td>
<td>Transmission bit</td>
</tr>
<tr>
<td>[45:40]</td>
<td>6</td>
<td>x</td>
<td>Command index</td>
</tr>
<tr>
<td>[39:8]</td>
<td>32</td>
<td>x</td>
<td>Argument</td>
</tr>
<tr>
<td>[7:1]</td>
<td>7</td>
<td>x</td>
<td>CRC7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>End bit</td>
</tr>
</tbody>
</table>

Next to the command data there are command type (WAITRESP) bits controlling the command path state machine (CPSM). These bits also determine whether the command requires a response, and whether the response is short (48 bit) or long (136 bits) long, and if a CRC is present or not.

A response is a token that is sent from an addressed card or synchronously from all connected cards to the host as an answer to a previous received command. All responses are sent via the command line SDMMC_CMD. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type. Response tokens R1, R2, R3, R4, R5, and R6 have various
coding schemes, depending on their content. The general formats for the response tokens for SD-Memory cards, SDIO cards, and eMMC cards are shown in Table 217, Table 218 and Table 219.

A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value denoted by x in the tables below indicates a variable entry. Most responses, except some, are protected by a CRC. Every command code word is terminated by the end bit (always 1).

The response token data is stored in 5 registers, four containing the 32-bits card status, OCR register, argument or 127-bits CID or CSD register including internal CRC, and one register containing the 6-bits command index.

### Table 217. Short response with CRC token format

<table>
<thead>
<tr>
<th>Bit position</th>
<th>Width</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>1</td>
<td>0</td>
<td>Start bit</td>
</tr>
<tr>
<td>46</td>
<td>1</td>
<td>0</td>
<td>Transmission bit</td>
</tr>
<tr>
<td>[45:40]</td>
<td>6</td>
<td>x</td>
<td>Command index (or reserved 111111)</td>
</tr>
<tr>
<td>[39:8]</td>
<td>32</td>
<td>x</td>
<td>Argument</td>
</tr>
<tr>
<td>[7:1]</td>
<td>7</td>
<td>x</td>
<td>CRC7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>End bit</td>
</tr>
</tbody>
</table>

### Table 218. Short response without CRC token format

<table>
<thead>
<tr>
<th>Bit position</th>
<th>Width</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>1</td>
<td>0</td>
<td>Start bit</td>
</tr>
<tr>
<td>46</td>
<td>1</td>
<td>0</td>
<td>Transmission bit</td>
</tr>
<tr>
<td>[45:40]</td>
<td>6</td>
<td>x</td>
<td>Command index (or reserved 111111)</td>
</tr>
<tr>
<td>[39:8]</td>
<td>32</td>
<td>x</td>
<td>Argument</td>
</tr>
<tr>
<td>[7:1]</td>
<td>7</td>
<td>1111111 (reserved 111111)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>End bit</td>
</tr>
</tbody>
</table>

### Table 219. Long response with CRC token format

<table>
<thead>
<tr>
<th>Bit position</th>
<th>Width</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>135</td>
<td>1</td>
<td>0</td>
<td>Start bit</td>
</tr>
<tr>
<td>134</td>
<td>1</td>
<td>0</td>
<td>Transmission bit</td>
</tr>
<tr>
<td>[133:128]</td>
<td>6</td>
<td>1111111</td>
<td>Reserved</td>
</tr>
<tr>
<td>[127:1]</td>
<td>127:8</td>
<td>x</td>
<td>CID or CSD slices</td>
</tr>
<tr>
<td></td>
<td>7:1</td>
<td>x</td>
<td>CRC7 (included in CID or CSD)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>End bit</td>
</tr>
</tbody>
</table>

The command/response path operates in a half-duplex mode, so that either commands can be sent or responses can be received. If the CPSM is not in the Send state, the
SDMMC_CMD output is in the Hi-Z state. Data sent on SDMMC_CMD are synchronous with the SDMMC_CK according the NEGEDGE register bit see Figure 158.

The command and short response with CRC, the CRC generator calculates the CRC checksum for all 40 bits before the CRC code. This includes the start bit, transmission bit, command index, and command argument (or card status).

For the long response the CRC checksum is calculated only over the 120 bits of R2 CID or CSD. Note that the start bit, transmission bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7-bit value:
\[
\text{CRC}[6:0] = \text{remainder } \left(\frac{(M(x) \cdot x^7)}{G(x)}\right)
\]
\[
G(x) = x^7 + x^3 + 1
\]
\[
M(x) = (\text{first bit}) \cdot x^n + (\text{second bit}) \cdot x^{n-1} + \ldots + (\text{last bit before CRC}) \cdot x^0
\]
Where \( n = 39 \) or 119.

The CPSM can send a number of specific commands to handle various operating modes when CPSMEN is set, see Table 220.

Table 220. Specific Commands overview

<table>
<thead>
<tr>
<th>VS(N)</th>
<th>BOOTEN</th>
<th>BOOTMOD</th>
<th>CMDTRAN</th>
<th>WAITPEND</th>
<th>CMDSTOP</th>
<th>WAITINT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 x x x x x x</td>
<td>0 1 x x x x</td>
<td>0 1 1 x x x</td>
<td>0 0 1 x x x</td>
<td>0 0 0 1 x x x</td>
<td>0 0 0 0 1 1 x</td>
<td>0 0 0 0 1 0 x</td>
<td>0 0 0 0 0 1 x</td>
</tr>
</tbody>
</table>
The command/response path implements the status flags and associated clear bits shown in Table 221:

### Table 221. Command path status flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMDSENT</td>
<td>Set at the end of the command without response. (CPSM moves from Send to Idle)</td>
</tr>
<tr>
<td>CMDREND</td>
<td>Set at the end of the command response when the CRC is OK. (CPSM moves from Receive to Idle)</td>
</tr>
<tr>
<td>CCRCFAIL</td>
<td>Set at the end of the command response when the CRC is FAIL. (CPSM moves from Receive to Idle)</td>
</tr>
<tr>
<td>CTIMEOUT</td>
<td>Set after the command when no response start bit received before the timeout. (CPSM moves from Wait to Idle)</td>
</tr>
<tr>
<td>CKSTOP</td>
<td>Set after the voltage switch (VSWITCHEN = 1) command response when the CRC is OK and the SDMMC_CK is stopped. (no impact on CPSM)</td>
</tr>
<tr>
<td>VSWEND</td>
<td>Set after the voltage switch (VSWITCH = 1) timeout of 5 ms + 1 ms. (no impact on CPSM)</td>
</tr>
<tr>
<td>CPSMACT</td>
<td>Command transfer in progress. (CPSM not in Idle state)</td>
</tr>
</tbody>
</table>

The command path error handling is shown in Table 222:

### Table 222. Command path error handling

<table>
<thead>
<tr>
<th>Error</th>
<th>CPSM state</th>
<th>Cause</th>
<th>Card action</th>
<th>Host action</th>
<th>CPSM action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>Wait</td>
<td>No start bit in time</td>
<td>Unknown</td>
<td>Reset or cycle power card(^1)</td>
<td>Move to Idle</td>
</tr>
<tr>
<td>CRC status</td>
<td>Receive</td>
<td>Negative status</td>
<td>Command ignored</td>
<td>Resend command(^1)</td>
<td>Move to Idle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transmission error</td>
<td>Command accepted</td>
<td>Resend command(^1)</td>
<td></td>
</tr>
</tbody>
</table>

\(^1\) When CMDTRANS is set, also a stop_transmission command must be send to move the DPSM to Idle.

### Data path

The data path subunit transfers data on the SDMMC_D[7:0] lines to and from cards. The data transmit path is clocked on the SDMMC_CK and sends data to the card. The data receive path is clocked on the sdmmc_rx_ck and receives data from the card. Figure 162 shows the data path block diagram.
The card data bus width can be programmed in the clock control register bits WIDBUS. The supported data bus width modes are:

- If the wide bus mode is not enabled, only one bit is transferred over SDMMC_D0.
- If the 4-bit wide bus mode is enabled, data is transferred at four bits over SDMMC_D[3:0].
- If the 8-bit wide bus mode is enabled, data is transferred at eight bits over SDMMC_D[7:0].

Next to the data bus width the data sampling mode can be programmed in the clock control register bit DDR. The supported data sampling modes are:

- Single data rate signaling (SDR), data is clocked on the rising edge of the clock.
- Double data rate signaling (DDR), data is clocked on the both edges of the clock. DDR mode is only supported in wide bus mode (4-bit wide and 8-bit wide).

Note: The data sampling mode only applies to the SDMMC_D[7:0] lines. (not applicable to the SDMMC_CMD line.)
In DDR mode, data is sampled on both edges of the SDMMC_CK according to the following rules, see also Figure 163 and Figure 164:

- On the rising edge of the clock odd bytes are sampled.
- On the falling edge of the clock even bytes are sampled.
- Data payload size is always a multiple of 2 Bytes.
- Two CRC16 are computed per data line
  - Odd bits CRC16 clocked on the falling edge of the clock.
  - Even bits CRC16 clocked on the rising edge of the clock.
- Start, end bits and idle conditions are full cycle.
- CRC status / boot acknowledgment and busy signaling are full cycle and are only sampled on the rising edge of the clock.

In DDR mode the SDMMC_CK clock division must be >= 2.

**Figure 163. DDR mode data packet clocking**

<table>
<thead>
<tr>
<th>SDMMC_CK</th>
<th>SDMMC_D3</th>
<th>SDMMC_D2</th>
<th>SDMMC_D1</th>
<th>SDMMC_D0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>start</td>
<td>start</td>
<td>start</td>
<td>start</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Byte 2</td>
<td>Byte 4</td>
<td>Byte n</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>end</td>
<td>end</td>
<td>end</td>
<td>end</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data odd</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data even</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRC odd</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRC even</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 164. DDR mode CRC status / boot acknowledgment clocking**

- Send: the DPSM moves to the Wait_S state. If there is data in the transmit FIFO, the DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- Receive: the DPSM moves to the Wait_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.
For boot operation with acknowledgment the DPSM moves to the Wait_Ack state and waits for the boot acknowledgment before moving to the Wait_R state.

The DPSM operates at SDMMC_CK. The DPSM has the following states, as shown in Figure 165. When ever the DPSM is active, i.e. not in the Idle state, the DPSMACT bit is set.

**Figure 165. Data path state machine (DPSM)**

- **Idle** state: the data path is inactive, and the SDMMC_D[7:0] outputs are according the PWRCTRL setting. The DPSM is activated either by sending a command with CMDTRANS bit set or by setting the DTEN bit, or by detecting Busy on SDMMC_D0 (that is, after a command with R1b response).

  When not busy, the DPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit), loads the data counter with a new (DATALENGTH) value and:
  - When the data direction bit (DTDIR) indicates send, moves to the Wait_S.
  - When the data direction bit (DTDIR) indicates receive, moves to the Wait_R when BOOTACKEN register bit is clear.
    - Wait_B when BOOTACKEN register bit is set and start the acknowledgment timeout.

  When busy the DPSM keeps the SDMMC_CK clock active and move to the Busy state.
Note: **DTEN must not be used to start data transfer with SD, SDIO and eMMC cards.**

- **Wait_Ack** state: the data path waits for the boot acknowledgment token.
  - The DPSM moves to the Wait_R state if it receives an error free acknowledgment before a timeout.
  - When a pattern different from the acknowledgment is received an acknowledgment status error is generated, and the ack fail status flag (ACKFAIL) is set. The DPSM stays in Wait_Ack.
  - If it reaches a timeout (ACKTIME) before it detects a start bit, it sets the timeout status flag (ACKTIMEOUT). The DPSM stays in Wait_Ack.
  - When the CPSM Abort signal is set it moves to the Idle state and sets the DABORT flag.

- **Wait_R** state: the data path, if the data counter is not zero and data is not hold, waits for a start bit on SDMMC_D[n:0]. If the data counter is zero or data is hold, wait for the FIFO to be empty.
  - In block mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DBLOCKSIZE.
  - In SDIO multibyte mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DATALENGTH.
  - In stream mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data counter with DATALENGTH.
  - if the data counter (DATACOUNT) equals zero (end of data) the DPSM moves to the Idle state when the receive FIFO is empty and the DATAEND flag is set.
  - If it reaches a timeout (DATATIME) before it detects a start bit, it sets the timeout status flag (DTIMEOUT) and the DPSM stays in the Wait_R state.
  - If the CPSM Abort signal is set:
    - If DATACOUNT > 0, the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST, and sets the DABORT flag.
    - If DATACOUNT is zero normal operation is continued, there is no DABORT flag since the transfer has completed normally.
  - if the DTHOLD bit is set:
    - When DATACOUNT > 0, the DPSM moves to the Idle state when the receive FIFO is empty and when IDMAEN = 0 reset with FIFORST, and issues the DHOLD flag. When holding the timeout is disabled. When an CPSM Abort signal is received during holding, the transfer is aborted.
- When DATACOUNT = 0, the transfer is completed normally and there is no DHOLD flag.
  - When DPSM has been started with DTEN, after an error (DTIMEOUT) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **R_W** state: the data path Read Wait the bus.
  - The DPSM moves to the Wait_R state when the Read Wait stop bit (RWSTOP) is set, and start the receive timeout.
  - If the CPSM Abort signal is set, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then moves to the Idle state and sets the DABORT flag.

- **Receive** state: the data path receives serial data from a card. Pack the data in bytes and written it to the data FIFO. Depending on the transfer mode selected in the data control register (DTMODE), the data transfer mode can be either block or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are received, the DPSM waits until it receives the CRC code.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are received, the DPSM waits until it receives the CRC code.

  a) If the received CRC code matches the internally generated CRC code, the DPSM moves to the
    - R_W state when RWSTART = 1 and DATACOUNT > zero, the DBCKEND flag is set.
    - Wait_R state otherwise.

  b) If the received CRC code fails the internally generated CRC code any further data reception is prevented.
    - When not all data has been received (DATACOUNT > 0), the CRC fail status flag (DCRCFAIL) is set and the DPSM stays in the Receive state.
    - When all data has been received (DATACOUNT = 0), wait for the FIFO to be empty after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.

  - In stream mode, the DPSM receives data while the data counter DATACOUNT > 0. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the Wait_R state.

  - When a FIFO overrun error occurs, the DPSM sets the FIFO overrun error flag (RXOVERR) and any further data reception is prevented. The DPSM stays in the Receive state.

  - When an CPSM Abort signal is received:
    - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted. The remaining data in the shift register is written to the data FIFO, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and the DABORT flag is set.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT=0, the transfer is completed normally. The DPSM stays in the Receive state no DABORT flag is generated.

  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or RXOVERR) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.
- **Wait_S** state: the data path waits for data to be available from the FIFO.
  - If the data counter DATACOUNT > 0, waits until the data FIFO empty flag (TXFIFOE) is de-asserted and DTHOLD is not set, and moves to the Send state.
  - If the data counter (DATACOUNT) = 0 the DPSM moves to the Idle state.
    - When DTHOLD is disabled, the DATAEND flag is set.
    - When DTHOLD is enabled, the DHOLD flag is set.
  - When DTHOLD is set and the DATACOUNT > 0
    - When IDMA is enabled, the DBCKEND flag is set and subsequently the FIFO is flushed, furthermore the DPSM moves to the Idle state and the DHOLD flag is set.
    - When IDMA is disabled the DBCKEND flag is set. Wait for the FIFO to be reset by software with FIFORST, then DPSM moves to the Idle state and issues the DHOLD flag.
  - When DTHOLD is set and DATACOUNT = 0 the transfer is completed normally.
  - When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and sets the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT=0, normal operation is continued, there is no DABORT flag since the transfer has completed normally.

  *Note: The DPSM remains in the Wait_S state for at least two clock periods to meet the N_WR timing requirements, where N_WR is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.*

- **Send** state: the DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block, SDIO multibyte or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  - In stream mode, the DPSM sends data to a card while the data counter DATACOUNT > 0. When the data counter reaches zero moves to the Busy state and start the transmit timeout.
  Before sending the last stream Byte according to DATACOUNT, the DPSM issues a trigger on the send CMD signal. This signal is used by the CPSM to sent any pending command. (i.e. CMD12 Stop Transmission command)
  - If a FIFO underrun error occurs, the DPSM sets the FIFO underrun error flag (TXUNDERR). The DPSM stays in the Send state.
  - When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the transfer with DATACOUNT=0, the transfer is aborted. The DPSM sends a last data bit followed by an end bit. The FIFO is disabled/flushed, and the DPSM moves to the Busy state to wait for not busy before setting the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer
with DATACOUNT=0, the transfer is completed normally, there is no DABORT flag.

- **Busy** state: the DPSM waits for the CRC status token when expected, and wait for a not busy signal:
  - If a CRC status token is expected and indicate "non-erroneous transmission" or when there is no CRC expected:
    - it moves to the Wait_S state when SDMMC_D0 is not low (the card is not busy).
    - When the card is busy SDMMC_D0 is low it remains in the Busy state.
  - If a CRC status token is expected and indicates "erroneous transmission".
    - When not all data has been send (DATACOUNT > 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set. The FIFO is disabled/flushed and the DPSM stays in the Busy state.
    - When all data has been send (DATACOUNT = 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - If a CRC status (Ncrc) timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - If a busy timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - When receiving the CPSM Abort signal in the Busy state:
    - If the CPSM Abort signal is received before the 2 last bits of the CRC response with DATACOUNT > 0, the data transfer is aborted. The DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
    - If the CPSM Abort signal is received during or after the 2 last bits of the CRC response when DATACOUNT = 0 or when no CRC is expected and DATACOUNT = 0 and there has been no DTIMEOUT error, the DPSM stays in the Busy state no DABORT flag is generated, since the transfer may completed normally.
    - If the CPSM Abort signal is received when a DTIMEOUT error has occurred the DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
  - When entering the Busy state due to an abort in the Send state, the DPSM waits for not busy before moving to the Idle state and the DABORT flag is set.
  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or DTIMEOUT) the DPSM moves to the Idle state when the FIFO is reset.
  - When the DPSM has been started due to Busy on SDMMC_D0, waits for not busy after which the Busy end status flag (BUSYD0END) is set and the DPSM moves to the Idle state.
The data timer (DATATIME) is enabled when the DPSM is in the Wait_R or Busy state 2 cycles after the data block end bit, or data read command end bit, or R1b response, and generates the data timeout error (DTIMEOUT):

- When transmitting data, the timeout occurs
  - when a CRC status is expected and no start bit is received withing 8 SDMMC_CK cycles, the DTIMEOUT flag is set.
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.

- When receiving data, the timeout occurs
  - when there is still data to be received DATACOUNT > 0 and no start bit is received before the programmed timeout period, the DTIMEOUT flag is set.

- After a R1b response, the timeout occurs
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.

When DATATIME = 0,

- In receive the start bit must be present 2 cycles after the data block end bit or data read command end bit.
- In transmit busy is timed out 2 cycles after the CRC token end bit or stream data end bit.
- After a R1b response busy is timed out 2 cycles after the response end bit.

Data can be transferred from the card to the host (transmit, send) or vice versa (receive). Data are transferred via the SDMMC_Dn data lines, they are stored in a FIFO.

### Table 223. Data token format

<table>
<thead>
<tr>
<th>Description</th>
<th>Start bit</th>
<th>Data(1)</th>
<th>CRC16</th>
<th>End bit</th>
<th>DTMODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block data</td>
<td>0</td>
<td>(DBLOCKSIZE, DATALENGTH)</td>
<td>yes</td>
<td>1</td>
<td>00</td>
</tr>
<tr>
<td>SDIO multibyte</td>
<td>0</td>
<td>(DATALENGTH)</td>
<td>yes</td>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td>eMMC stream</td>
<td>0</td>
<td>(DATALENGTH)</td>
<td>no</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

1. The total amount of data to transfer is given by DATALENGTH. Where for Block data the amount of data in each block is given by DBLOCKSIZE.

The data token format is selected with register bits DTMODE according.

The data path implements the status flags and associated clear bits shown in Table 224:

### Table 224. Data path status flags and clear bits

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATAEND TX</td>
<td>Set at the end of the complete data transfer when the CRC is OK and busy has finished and both DTHOLD = 0 and DATACOUNT = 0. (DPSM moves from Wait_S to Idle)</td>
</tr>
<tr>
<td>DATAEND RX</td>
<td>Set at the end of the complete data transfer when the CRC is OK and all data has been read, (DATACOUNT = 0 and FIFO is empty). (DPSM moves from Wait_R to Idle)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boot</td>
<td>When DATATIME = 0,</td>
</tr>
</tbody>
</table>

948/3119 RM0481 Rev 1
## Table 224. Data path status flags and clear bits (continued)

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCRCFAIL</td>
<td>TX: Set at the end of the CRC when FAIL and busy has finished. (DPSM stay in Busy when there is still data to send and wait for CPSM Abort) (DPSM moves from Busy to Idle when all data has been sent) or DPSM has been started with DTEN.</td>
</tr>
<tr>
<td></td>
<td>RX: Set at the end of the CRC when FAIL and FIFO is empty. (DPSM stays in Receive when there is still data to be received and wait for CPSM Abort) (DPSM moves from Receive to Idle when all data has been received or DPSM has been started with DTEN)</td>
</tr>
<tr>
<td></td>
<td>Boot: Set at the end of the CRC when FAIL and FIFO is empty. (DPSM moves from Receive to Idle when all data has been received or DPSM has been started with DTEN)</td>
</tr>
<tr>
<td>ACKFAIL</td>
<td>Boot: Set at the end of the boot acknowledgment when fail. (DPSM stays in Wait_Ack and wait for CPSM Abort)</td>
</tr>
<tr>
<td>DTIMEOUT</td>
<td>TX: Set when no CRC token start bit received within Ncrc, or no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle)</td>
</tr>
<tr>
<td></td>
<td>RX: Set when no start bit received before the timeout. (DPSM stays in Wait_R and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle)</td>
</tr>
<tr>
<td></td>
<td>Boot: Set when no start bit received before the timeout. (DPSM stays in Wait_Ack and wait for CPSM Abort)</td>
</tr>
<tr>
<td>ACKTIMEOUT</td>
<td>Boot: Set when no start bit received before the timeout. (DPSM stays in Wait_Ack and wait for CPSM Abort)</td>
</tr>
<tr>
<td>DBCKEND</td>
<td>TX: When DTHOLD = 1 and IDMAEN = 0: Set at the end of data block transfer when the CRC is OK and busy has finished, when data transfer is not complete (DATACOUNT &gt;0). (DPSM moves from Busy to Wait_S)</td>
</tr>
<tr>
<td></td>
<td>RX: When RWSTART = 1: Set at the end of data block transfer when the CRC is OK, when data transfer is not complete (DATACOUNT &gt; 0). (DPSM moves from Receive to R_W)</td>
</tr>
<tr>
<td></td>
<td>Boot: When RWSTART = 1: Set at the end of data block transfer when the CRC is OK and all data has been read (FIFO is empty), when data transfer is not complete (DATACOUNT &gt;0). (DPSM moves from Wait_R to Idle)</td>
</tr>
<tr>
<td>DHOLD</td>
<td>TX: When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and busy has finished. (DPSM moves from Wait_S to Idle)</td>
</tr>
<tr>
<td></td>
<td>RX: When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and all data has been read (FIFO is empty), when data transfer is not complete (DATACOUNT &gt;0). (DPSM moves from Wait_R to Idle)</td>
</tr>
<tr>
<td>DABORT</td>
<td>CMD R1b: When CPSM Abort event has been sent by the CPSM and busy has finished. (DPSM moves from Busy to Idle)</td>
</tr>
<tr>
<td></td>
<td>TX: When CPSM Abort event has been sent by the CPSM before the 2 last bits of the transfer. (DPSM moves from any state to Idle)</td>
</tr>
<tr>
<td></td>
<td>RX: When CPSM Abort event has been sent by the CPSM before the 2 last bits of the transfer. (DPSM moves from any state to Idle)</td>
</tr>
<tr>
<td>BUSYD0END</td>
<td>CMD R1b: Set after the command response when end of busy before the timeout. (DPSM moves from Busy to Idle)</td>
</tr>
<tr>
<td>DPSMACT</td>
<td>Data transfer in progress. (DPSM not in Idle state)</td>
</tr>
</tbody>
</table>
The data path error handling is shown in Table 225:

<table>
<thead>
<tr>
<th>Error</th>
<th>DPSM state</th>
<th>Cause</th>
<th>Card action</th>
<th>Host action</th>
<th>DPSM action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>Wait_Ack</td>
<td>No Ack in time</td>
<td>unknown</td>
<td>Card cycle power</td>
<td>Stay in Wait_Ack (reset the SDMMC with the RCC.SDMMCCxRST register bit)</td>
</tr>
<tr>
<td></td>
<td>Wait_R</td>
<td>No start bit in time</td>
<td>unknown</td>
<td>Stop data reception</td>
<td>Send stop transmission command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Send stop data reception command</td>
<td></td>
</tr>
<tr>
<td>Busy</td>
<td></td>
<td>Busy too long (due to data transfer)</td>
<td>unknown</td>
<td>Stop data reception</td>
<td>Send stop transmission command</td>
</tr>
<tr>
<td>CRC</td>
<td>Receive</td>
<td>transmission error</td>
<td>Send further data</td>
<td>Stop data reception</td>
<td>On CPSM Abort move to Idle</td>
</tr>
<tr>
<td>CRC status</td>
<td>Busy</td>
<td>Negative status</td>
<td>Ignore further data</td>
<td>Stop data transmission</td>
<td>On CPSM Abort move to Idle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>transmission error</td>
<td>wait for further data</td>
<td>Send stop transmission command</td>
<td></td>
</tr>
<tr>
<td>Ack status</td>
<td>Wait_Ack</td>
<td>transmission error</td>
<td>Send boot data</td>
<td>Stop boot procedure</td>
<td>On CPSM Abort move to Idle</td>
</tr>
<tr>
<td>Overrun</td>
<td>Receive</td>
<td>FIFO full</td>
<td>Send further data</td>
<td>Stop data reception</td>
<td>On CPSM Abort move to Idle</td>
</tr>
<tr>
<td>Underrun</td>
<td>Send</td>
<td>FIFO empty</td>
<td>Receive further data</td>
<td>Stop data transmission</td>
<td>On CPSM Abort move to Idle</td>
</tr>
</tbody>
</table>

**Data FIFO**

The data FIFO (first-in-first-out) subunit contains the transmit and receive data buffer. A single FIFO is used for either transmit or receive as selected by the DTDIR bit. The FIFO contain a 32-bit wide, 16-word deep data buffer and control logic. Because the data FIFO operates in the AHB clock domain (sdmmc_hclk), all signals from the subunits in the SDMMC clock domain (SDMMC_CK/sdmmc_rx_ck) are resynchronized.
The FIFO can be in one of the following states:

- The transmit FIFO refers to the transmit logic and data buffer when sending data out to the card. (DTDIR = 0)
- The receive FIFO refers to the receive logic and data buffer when receiving data in from the card. (DTDIR = 1)

The end of a correctly completed SDMMC data transfer from the FIFO is indicated by the DATAEND flags driven by the data path subunit. Any incorrect (aborted) SDMMC data transfer from the FIFO is indicated by one of the error flags (DCRCFAIL, DTIMEOUT, DABORT) driven by the data path subunit, or one of the FIFO error flags (TXUNDERR, RXOVERR) driven by the FIFO control.

The data FIFO can be accessed in the following ways, see Table 226.

### Table 226. Data FIFO access

<table>
<thead>
<tr>
<th>Data FIFO access</th>
<th>IDMAEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>From firmware via AHB slave interface</td>
<td>0</td>
</tr>
<tr>
<td>From IDMA via AHB master interface</td>
<td>1</td>
</tr>
</tbody>
</table>

Transmit FIFO:

Data can be written to the transmit FIFO when the DPSM has been activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface. The transmit FIFO is accessible via sequential addresses. The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out.

The transmit FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   - For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in transmit mode (DTDIR = 0).
   - Configures the FIFO in transmit mode.
3. Enable the data transfer
   - either by sending a command from the CPSM with the CMDTRANS bit set
   - or by setting DTEN bit
4. When (DPSMACT = 1) write data to the FIFO.
   - The DPSM stays in the Wait_S state until FIFO is full (TXFIFOF = 1), or the number indicated by DATALENGTH.
- The SDMMC keeps sending data as long as FIFO is not empty, hardware flow control during data transfer is used to prevent FIFO underrun.

5. Write data to the FIFO.
   - When the FIFO is handled by software, wait until the FIFO is half empty (TXFIFOHE flag), write data to the FIFO until FIFO is full (TXFIFOF = 1), or last data has been written.
   - When the FIFO is handled by the IDMA, the IDMA transfers the FIFO data.

6. When last data has been written wait for end of data (DATAEND flag)
   - SDMMC has completely sent all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer error or transfer hold when IDMAEN = 0, firmware must stop writing to the FIFO and flush and reset the FIFO with the FIFORST register bit.

The transmit FIFO status flags are listed in Table 227.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXFIFOF</td>
<td>Set to high when all transmit FIFO words contain valid data.</td>
</tr>
<tr>
<td>TXFIFOE</td>
<td>Set to high when the transmit FIFO does not contain valid data.</td>
</tr>
<tr>
<td>TXFIFOHE</td>
<td>Set to high when half or more transmit FIFO words are empty.</td>
</tr>
<tr>
<td>TXUNDERR</td>
<td>Set to high when an underrun error occurs. This flag is cleared by writing to the SDMMC Clear register.</td>
</tr>
</tbody>
</table>

Receive FIFO:

Data can be read from the receive FIFO when the DPSM is activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface. When the data path subunit receives a word of data, it drives the data on the write databus. The write pointer is incremented after the write operation completes. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer is driven onto the read databus. The receive FIFO is accessible via sequential addresses.

The receive FIFO is handled in the following way:
1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in receive mode (DTDIR = 1).
   – Configures the FIFO in receive mode.
3. Enable the DPSM transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit.
4. When (DPSMACT = 1) the FIFO is ready to receive data.
   – The DPSM writes the received data to the FIFO.
   - The SDMMC keeps receiving data as long as FIFO is not full, hardware flow control during the data transfer is used to prevent FIFO overrun.
5. Read data from the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half full (RXFIFOHF flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
   - When last data has been received, read data from the FIFO until FIFO is empty (DATAEND = 1).
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO data.
6. SDMMC has completely received all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer hold when IDMAEN = 0, the firmware must read the remaining data until the FIFO is empty and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

In case of a data transfer error when IDMAEN = 0, the firmware must stop reading the FIFO and flush and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

The receive FIFO status flags are listed in Table 228.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXFIFOF</td>
<td>Set to high when all receive FIFO words contain valid data</td>
</tr>
<tr>
<td>RXFIFOE</td>
<td>Set to high when the receive FIFO does not contain valid data.</td>
</tr>
<tr>
<td>RXFIFOHF</td>
<td>Set to high when half or more receive FIFO words contain valid data.</td>
</tr>
<tr>
<td>RXOVERR</td>
<td>Set to high when an overrun error occurs. This flag is cleared by writing to the SDMMC Clear register.</td>
</tr>
</tbody>
</table>

**CLKMUX unit**

The CLKMUX selects the source for clock sdmmc_rx_ck to be used with the received data and command response. The receive data clock source can be selected by the clock control register bit SELCLKRX, between:

- sdmmc_io_in_ck bus master main feedback clock.
- SDMMC_CKIN external bus feedback clock.
- sdmmc_fb_ck bus tuned feedback clock.

The sdmmc_io_in_ck is selected when there is no external driver, with DS and HS.
The SDMMC_CKIN is selected when there is an external driver with SDR12, SDR25, SDR50 and DDR50.

The sdmmc_fb_ck clock input must be selected when the DLYB block on the device is used with SDR104, HS200 and optionally with SDR50 and DDR50 modes.

The sdmmc_rx_ck source must be changed when the CPSM and DPSM are in the Idle state.

### 24.5.5 SDMMC AHB slave interface

The AHB slave interface generates the interrupt requests, and accesses the SDMMC adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt logic.

#### SDMMC FIFO

The FIFO access is restricted to word access only:

- In transmit FIFO mode
  - Data are written to the FIFO in words (32-bits) until all data according to \( \text{DATALENGTH} \) has been transferred. When the \( \text{DATALENGTH} \) is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are written with a word transfer.

- In receive FIFO mode
  - Data are read from the FIFO in words (32-bits) until all data according to \( \text{DATALENGTH} \) has been transferred. When the \( \text{DATALENGTH} \) is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are read with a word transfer padded with 0 value bytes.

When accessing the FIFO with half word or byte accesses an AHB bus fault is generated.

#### SDMMC interrupts

The interrupt logic generates an interrupt request signal that is asserted when at least one of the unmasked status flags is active. A mask register is provided to allow selection of the conditions that generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set. Some status flags require an implicit clear in the clear register.

### 24.5.6 SDMMC AHB master interface

The AHB master interface is used to transfer the data between a memory and the FIFO using the SDMMC IDMA.
SDMMC IDMA

Direct memory access (DMA) is used to provide high-speed transfer between the SDMMC FIFO and the memory. The AHB master optimizes the bandwidth of the system bus. The SDMMC internal DMA (IDMA) provides one channel to be used either for transmit or receive.

The IDMA is enabled by the IDMAEN bit and supports burst transfers of 8 beats.

- **In transmit burst transfer mode:**
  - Data are fetched in burst from memory whenever the FIFO is empty for the number of burst transfers, until all data according DATALENGTH has been transferred. When the DATALENGTH is not an integer multiple of the burst size the remaining, smaller then burst size data is transferred using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are fetched with a word transfer.

- **In receive burst transfer mode:**
  - Data are stored in burst into memory whenever the FIFO contains the number of burst transfers, until all data according DATALENGTH has been transferred. When the DATALENGTH is not an integer multiple of the burst transfer the remaining, smaller then burst size data, is transferred using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are stored with halfword and or byte transfers.

In addition the IDMA provides the following channel configurations selected by bit IDMABMODE:

- single buffered channel
- linked list channel

**Single buffered channel**

In single buffer configuration the data at the memory side is accessed in a linear matter starting from the base address IDMABASE. When the IDMA has finished transferring all data the and the DPSM has completed the transfer the DATAEND flag is set.

**Linked list channel**

In linked list configuration, IDMAMODE = 1, the data at the memory side is subsequently accessed from linked buffers, located at base address IDMABASE. The size of the memory buffers is defined by IDMABSIZE. The buffer size must be an integer multiple of the burst size. The bit ULA is used to indicate if a new linked list buffer configuration has to be loaded from the linked list table. A new linked list configuration is loaded when the ULA bit for the current linked list item is set.

The first linked list item configuration is programmed by firmware directly in the SDMMC registers.

When the IDMA has finished transferring all the data of one linked list buffer, according IDMABSIZE, and when the linked list item ULA bit is set, the IDMA loads the new linked list item from the linked list table, and continues transferring data from the next linked list buffer. When the IDMA has finished transferring all data, according IDMABSIZE and ULA, and the DPSM has completed the transfer, according DATALENGHT, the DATAEND flag is set.
In the following cases, the linked list provides more buffer space than the data to transfer which means the current linked list buffer data has not completely be transfered:

- the ULA bit is set, and all SDMMC data according DATALENGTH has been transfered (DATAEND flag)
- a transfer error (DCRCFAIL when DATACOUNT > 0, RXOVERR, TXUNDERR) occurs
- a transfer is hold (DTHOLD)

In all above cases, the IDMA linked list is stopped and the FIFO is flushed/reset. Before starting or restarting a new SDMMC transfer, the software must initialize a new linked list with correct IDMABASE and IDMABSIZE.

When a IDMA transfer error occurs (see Section : IDMA transfer error management) or when the linked list does not provide sufficient buffer space:

- the linked list ends with ULA = 0 and all last linked list buffer data has been transfered, and not all SDMMC data according DATALENGTH has been transfered. The SDMMC transfer is stopped and an IDMA transfer error is generated (see Section : IDMA transfer error management).

For a given linked list item, the base address is given by the linked list base IDMABA register value plus the linked list offset IDMALA register value.

The content of each linked list item can be specified by the ULS bit, which makes possible to optionally load the IDMABSIZ, resulting in a 3-word linked list structure. When the IDMABSIZ is not to be loaded (i.e. fixed size buffers) a compacted reduced 2-word linked list structure can be used containing only the IDMABASER and the IDMALAR values.

For a given linked list item, the base address is given by the linked list base IDMABA register value plus the linked list offset IDMALA register value.

The content of each linked list item can be specified by the ULS bit, which makes possible to optionally load the IDMABSIZ, resulting in a 3-word linked list structure. When the IDMABSIZ is not to be loaded (i.e. fixed size buffers) a compacted reduced 2-word linked list structure can be used containing only the IDMABASER and the IDMALAR values.

**Figure 167. Linked list structures**

There is no restriction on mixing both linked list item structures in a single list, this enables the IDMABSIZ to be updated only when needed.

Whenever a linked list buffer has been transfered and the current buffer ULA = 1, an end-of-linked-list-buffer-transfer-complete interrupt (IDMABTC) may be generated (if interrupt is enabled).
Linked list acknowledgment

In the case where software dynamically updates the linked list, during the SDMMC transfer, the availability of a new linked list buffer can be acknowledged by the acknowledge buffer ready (ABR) bit.

When ABR acknowledges that the new linked list buffer is ready, the IDMA continues transferring data from the new linked list buffer.

When ABR indicates that the new linked list buffer is not ready, an IDMA transfer error is generated (see Section : IDMA transfer error management). Depending when the IDMA transfer error occurs, it normally causes the generation of an TXUNDERR or RXOVERR error. When a linked list buffer is not acknowledged in time the SDMMC transfer is stopped.

The ABR information is “don’t care” when starting the linked list from software programmed register information. The first linked list buffer must be ready to be used before starting the SDMMC transfer.

IDMA transfer error management

An IDMA transfer error can occur:

- When reading or writing a reserved address space (for data or linked list information).
- When there is no more linked list buffer space to store received SDMMC data.
- When all linked list buffer data has been transferred and still more SDMMC data needs to be sent.
- When the availability of a linked list buffer is not acknowledged.

On a IDMA transfer error subsequent IDMA transfers are disabled and an IDMATE flag is set and hardware flow control is disabled. Depending when the IDMA transfer error occurs, it normally causes the generation of a TXUNDERR or RXOVERR error.

The behavior of the IDMATE flag depend on when the IDMA transfer error occurs during the SDMMC transfer:

- An IDMA transfer error is detected before any SDMMC transfer error (TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT):
  - The IDMATE flag is set at the same time as the SDMMC transfer error flag.
  - The TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT interrupt is generated.
- An IDMA transfer error is detected during a STOP_TRANSNMISSION command:
  - The IDMATE flag is set at the same time as the DABORT flag.
  - The DABORT interrupt is generated.
- An IDMA transfer error is detected at the end of the SDMMC transfer (DHOLD, or DATAEND):
  - The IDMATE flag is set at the end of the SDMMC transfer.
  - A SDMMC transfer end interrupt is generated and a DHOLD or DATAEND flag is set.

The IDMATE is generated on an other SDMMC transfer interrupt (TXUNDERR, RXOVERR, DCRCFAIL, DTIMEOUT, DABORT, DHOLD, or DATAEND).
### 24.5.7 AHB and SDMMC_CK clock relation

The AHB must at least have 3x more bandwidth than the SDMMC bus bandwidth i.e. for SDR50 4-bit mode (50 Mbyte/s) the minimum sdmmc_hclk frequency is 37.5 MHz (150 Mbyte/s).

<table>
<thead>
<tr>
<th>SDMMC bus mode</th>
<th>SDMMC bus width</th>
<th>Maximum SDMMC_CK [MHz]</th>
<th>Minimum AHB clock [MHz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>eMMC DS</td>
<td>8</td>
<td>26</td>
<td>19.5</td>
</tr>
<tr>
<td>eMMC HS</td>
<td>8</td>
<td>52</td>
<td>39</td>
</tr>
<tr>
<td>eMMC DDR52</td>
<td>8</td>
<td>52</td>
<td>78</td>
</tr>
<tr>
<td>eMMC HS200</td>
<td>8</td>
<td>200</td>
<td>150</td>
</tr>
<tr>
<td>SD DS / SDR12</td>
<td>4</td>
<td>25</td>
<td>9.4</td>
</tr>
<tr>
<td>SD HS / SDR25</td>
<td>4</td>
<td>50</td>
<td>18.8</td>
</tr>
<tr>
<td>SD DDR50</td>
<td>4</td>
<td>50</td>
<td>37.5</td>
</tr>
<tr>
<td>SD SDR50</td>
<td>4</td>
<td>100</td>
<td>37.5</td>
</tr>
<tr>
<td>SD SDR104</td>
<td>4</td>
<td>208</td>
<td>78</td>
</tr>
</tbody>
</table>

### 24.6 Card functional description

#### 24.6.1 SD I/O mode

The following features are SDMMC specific operations:
- SDIO interrupts
- SDIO suspend/resume operation (write and read suspend)
- SDIO Read Wait operation by stopping the clock
- SDIO Read Wait operation by SDMMC_D2 signaling

<table>
<thead>
<tr>
<th>Operation mode</th>
<th>SDOEN</th>
<th>RWMOD</th>
<th>RWSTOP</th>
<th>RWSTART</th>
<th>DDIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt detection</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Suspend/Resume operation</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Read Wait SDMMC_CK clock stop (START)</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Read Wait SDMMC_CK clock stop (STOP)</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Read Wait SDMMC_D2 signaling (START)</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Read Wait SDMMC_D2 signaling (STOP)</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
**SD I/O interrupts**

To allow the SD I/O card to interrupt the host, an interrupt function is available on pin 8 (shared with SDMMC_D1 in 4-bit mode) on the SD interface. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level-sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the host or deasserted due to the end of the interrupt period. After the host has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card internal registers. The interrupt output of all SD I/O cards is active low and the application must provide external pull-up resistors on all data lines (SDMMC_D[3:0]).

In SD 1-bit mode pin 8 is dedicated to the interrupt function (IRQ), and there are no timing constraints on interrupts.

In SD 4-bit mode the host samples the level of pin 8 (SDMMC_D1/IRQ) into the interrupt detector only during the interrupt period. At all other times, the host interrupt ignores this value. The interrupt period begins when interrupts are enabled at the card and SDIOEN bit is set see register settings in Table 230.

In 4-bit mode the card can generate a synchronous or asynchronous interrupt as indicated by the card CCCR register SAI and EAI bits.

- Synchronous interrupt, require the SDMMC_CK to be active.
- Asynchronous interrupt, can be generated when the SDMMC_CK is stopped, 4 cycles after the start of the card interrupt period following the last data block.

The timing of the interrupt period is depended on the bus speed mode:
In DS, HS, SDR12, and SDR25 mode, selected by register bit BUSSPEED, the interrupt period is synchronous to the SD clock.

- The interrupt period ends at the next clock from the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set.
- The interrupt period resumes 2 SDMMC_CK after the completion of the data block.
- At the data block gap the interrupt period is limited to 2 SDMMC_CK cycles.

*Note:* DTEN must not be used to start data transfer with SD and eMMC cards.

**Figure 169. Synchronous interrupt period data read**

**Figure 170. Synchronous interrupt period data write**
In SDR50, SDR104, and DDR50, selected by register bit BUSSPEED, due to propagation delay from the card to host, the interrupt period is asynchronous.

- The card interrupt period ends after 0 to 2 SDMMC_CK cycles after the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set. At the host the interrupt period ends after the end bit of a command that transfers data block(s). A card interrupt issued in the 1 to 2 cycles after the command end bit are not detected by the host during this interrupt period.
- The card interrupt period resumes 2 to 4 SDMMC_CK after the completion of the last data block. The host resumes the interrupt period always 2 cycles after the last data block.
- There is NO interrupt period at the data block gap.

**Note:** *DTEN must not be used to start data transfer with SD and eMMC cards.*

**Figure 171. Asynchronous interrupt period data read**
When transferring Open-ended multiple block data and using DTMODE “block data transfer ending with STOP_TRANSMISSION command”, the SDMMC masks the interrupt period after the last data block until the end of the CMD12 STOP_TRANSMISSION command.

The interrupt period is applicable for both memory and I/O operations.

In 4-bit mode interrupts can be differentiated from other signaling according Table 231.

**Table 231. 4-bit mode Start, interrupt, and CRC-status Signaling detection**

<table>
<thead>
<tr>
<th>SDMMC data line</th>
<th>Start</th>
<th>Interrupt</th>
<th>CRC-status</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMMC_D0</td>
<td>0</td>
<td>1 or CRC-status</td>
<td>0</td>
</tr>
<tr>
<td>SDMMC_D1</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>SDMMC_D2</td>
<td>0</td>
<td>1 or Read Wait</td>
<td>X</td>
</tr>
<tr>
<td>SDMMC_D3</td>
<td>0</td>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

**SD I/O suspend and resume**

This function is NOT supported in SDIO version 4.00 or later.

Within a multifunction SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the eMMC/SD bus. To share access to the host among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the host can temporarily halt (suspend) a data transfer operation to one function or memory to free the bus for a higher-priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is restarted (resume) where it left off.

To perform the suspend/resume operation on the bus, the host performs the following steps:
1. Determines the function currently using the SDMMC_D[3:0] line(s)
2. Requests the lower-priority or slower transaction to suspend
3. Waits for the transaction suspension to complete
4. Begins the higher-priority transaction
5. Waits for the completion of the higher priority transaction
6. Restores the suspended transaction

The card receiving a suspend command responds with its current bus status. Only when the bus has been suspended by the card the bus status indicates suspension completed.

There are different suspend cases conditions:
- Suspend request accepted prior to the start of data transfer.
- Suspend request not accepted, (due to data being transfered at the same time), the host keeps checking the request until it is accepted. (data transfer has suspended)
- Suspend request during write busy.
- Suspend request with write multiple.
- Suspend request during Read Wait.

For the host to know if the bus has been released it must check the status of the suspend request, suspension completed.

When the bus status of the suspend request response indicates suspension completed, the card has released the bus. At this time the state of the suspended operation must be saved where after an other operation can start.

The suspend command must be sent with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

The hardware does not save the number of remaining data to be transfered when resuming the suspended operation. It is up to firmware to determine the data that has been transferred and resume with the correct remaining number of data bytes.

While receiving data from the card, the SDMMC can suspend the read operation after the read data block end (DPSM in Wait_R). After receiving the suspend acknowledgment response from the card the following steps must be taken by firmware:

1. The normal receive process must be stopped by setting DTHOLD bit.
   a) The remaining number of data bytes in the FIFO must be read until the receive FIFO is empty (RXFIFOE flag is set), and when IDMAEN = 0 the FIFO must be reset with FIFORST.
2. The confirmation that all data has been read from the FIFO, and that the suspend is completed is indicated by the DHOLD flag.
   a) The remaining number of data bytes (multiple of data blocks) still to be read when resuming the operation must be determined from the remaining number of bytes indicated by the DATACOUNT.

Note: When a DTIMEOUT flag occurs during the suspend procedure, this must be ignored.

To resume receiving data from the card, the following steps must be taken by firmware:
1. The remaining number of data bytes (multiple of data blocks) must be programmed in \textit{DATALENGTH}.
2. The DPSM must be configured to receive data in the DTDIR bit.
3. The resume command must be sent from the CPSM, with the CMDTRANS bit set and the CMDSUSPEND bit set, which ends the interrupt period when data transfer is resumed (response bit DF = 1) and enabled the DPSM, after which the card resumes sending data.

While sending data to the card, the SDMMC can suspend the write operation after the write data block CRC status end (DPSM in Busy). Before sending the suspend command to the card the following steps must be taken by firmware:
1. Enable DHOLD flag (and DBCKEND flag when IDMAEN = 0)
2. The DPSM must be prevented from start sending a new data block by setting DTHOLD.
3. When IDMAEN = 0: When receiving the DBCKEND flag the data transfer is stopped. Firmware can stop filling the FIFO, after which the FIFO must be reset with FIFORST. Any bytes still in the FIFO need to be rewritten when resuming the operation.
4. When receiving the DHOLD flag the data transfer is stopped. The remaining number of data bytes still to be written when resuming must be determined from the remaining number of bytes indicated by the DATACOUNT.
5. To suspend the card the suspend command must be sent by the CPSM with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

To resume sending data to the card, the following steps must be taken by firmware:
1. The remaining number of data bytes must be programmed in \textit{DATALENGTH}.
2. The DPSM must be configured for transmission with DTDIR set and enabled by having the CPSM send the resume command with the CMDTRANS bit set and the CMDSUSPEND bit set. This ends the interrupt period and start the data transfer. The DPSM either goes to the Wait\_S state when SDMMC\_D0 does not signal busy, or goes to the Busy state when busy is signaled.
3. When IDMAEN = 1: The IDMA needs to be reprogrammed for the remaining bytes to be transfered.
4. When IDMAEN = 0: Firmware must start filling the FIFO with the remaining data.

\textbf{SD I/O Read Wait}

There are 2 methods to pause the data transfer during the Block gap:
1. Stopping the SDMMC\_CK.
2. Using Read Wait signaling on SDMMC\_D2.

The SDMMC can perform a Read Wait with register settings according \textit{Table 230}.

Depending the SDMMC operation mode (DS, HS, SDR12, SDR25) or (SDR50, SDR104, DDR) each method has a different characteristic.

The timing for pause read operation by stopping the SDMMC\_CK for DS, HS, SDR12, and SDR25, the SDMMC\_CK may be stopped 2 SDMMC\_CK cycles after the end bit. When ready the host resumes by restarting clock, see \textit{Figure 173}.  

\textit{Table 230}
The timing for pause read operation by stopping the SDMMC_CK for SDR50, SDR104, and DDR50, the SDMMC_CK may be stopped minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, after the end bit. When ready the host resumes by restarting clock, see Figure 174. (In DDR50 mode the SDMMC_CK must only be stopped after the falling edge, when the clock line is low.)

In Read Wait SDMMC_CK clock stopping, when RWSTART is set, the DSPM stops the clock after the end bit of the current received data block CRC. The clock start again after writing 1 to the RWSTOP bit, where after the DPSM waits for a start bit from the card.

As SDMMC_CK is stopped, no command can be issued to the card. During a Read Wait interval, the SDMMC can still detect SDIO interrupts on SDMMC_D1.

The optional Read Wait signaling on SDMMC_D2 (RW) operation is defined only for the SD 1-bit and 4-bit modes. The Read Wait operation enables the host to signal a card that is reading multiple registers (IO_RW_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O device. To determine when a card supports the Read Wait protocol, the host must test capability bits in the internal card registers.

The timing for Read Wait with a SDMMC_CK less then 50MHz (DS, HS, SDR12, SDR25) is based on the interrupt period generated by the card on SDMMC_D1. The host by asserting SDMMC_D2 low during the interrupt period requests the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z, see Figure 175.
For SDR50, SDR104 with a SDMMC_CK more than 50MHz, and DDR50, the card treats the Read Wait request on SDMMC_D2 as an asynchronous event. The host by asserting SDMMC_D2 low after minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, request the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z. The host must raise SDMMC_D2 on the SDMMC_CK clock (see Figure 176).

In Read Wait SDMMC_D2 signaling, when RWSTART is set, the DPSM drives SDMMC_D2 after the end bit of the current received data block CRC. The Read Wait signaling on SDMMC_D2 is removed when writing 1 to the RWSTOP bit. The DPSM remains in R_W state for two more SDMMC_CK clock cycles to drive SDMMC_D2 to 1 for one clock cycle (in accordance with SDIO specification), where after the DPSM waits for a start bit from the card.

During the Read Wait signaling on SDMMC_D2 commands can be issued to the card. During the Read Wait interval, the SDMMC can detect SDIO interrupts on SDMMC_D1.

### 24.6.2 CMD12 send timing

CMD12 is used to stop/abort the data transfer, the card data transmission is terminated two clock cycles after the end bit of the Stop Transmission command.
Table 232. CMD12 use cases

<table>
<thead>
<tr>
<th>Data operation</th>
<th>Stop Transmission command CMD12</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMMC stream write</td>
<td></td>
<td>The data transfer is stopped/aborted by sending the Stop Transmission command.</td>
</tr>
<tr>
<td>SDMMC open ended multiple block write</td>
<td></td>
<td>The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.</td>
</tr>
<tr>
<td>SDMMC block write with predefined block count</td>
<td></td>
<td>The Stop Transmission command is not required at the end of this type of multiple block write. (sending the Stop Transmission command after the card has received the last block is regarded as an illegal command.) If the card detects an error, the host must abort the operation by sending the Stop Transmission command.</td>
</tr>
<tr>
<td>SDMMC stream read</td>
<td></td>
<td>The data transfer is stopped/aborted by sending the Stop Transmission command.</td>
</tr>
<tr>
<td>SDMMC open ended multiple block read</td>
<td></td>
<td>The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.</td>
</tr>
<tr>
<td>SDMMC block read with predefined block count</td>
<td></td>
<td>The Stop Transmission command is not required at the end of this type of multiple block read. (sending the Stop Transmission command after the card has transmitted the last block is regarded as an illegal command.) Transaction can be aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.</td>
</tr>
</tbody>
</table>

All data write and read commands can be aborted any time by a Stop Transmission command CMD12. The following data abort procedure applies during an ongoing data transfer:
1. Load CMD12 Stop Transmission command in registers and set the CMDSTOP bit.
   a) This causes the CPSM Abort signal to be generated when the command is sent to
      the DPSM.
2. Configure the CPSM to send a command immediately (clear WAITPEND bit).
   a) The card, when sending data, stops data transfer 2 cycles after the Stop
      Transmission command end bit.
      The card when no data is being sent, does not start sending any new data.
   b) The host, when sending data, sends one last data bit followed by an end bit after
      the Stop Transmission command end bit.
      The host when not sending data, does not start sending any new data.
3. When IDMAEN = 0, the FIFO need to be reset with FIFORST.
   a) When writing data to the card. On the CMDREND flag, firmware must stop writing
      data to the FIFO. Subsequently the FIFO must be reset with FIFORST, this flushes
      the FIFO.
   b) When reading data from the card. On the CMDREND flag, firmware must read the
      remaining data from the FIFO. Subsequently the FIFO must be reset with
      FIFORST.
4. When IDMAEN = 1, hardware takes care of the FIFO.
   a) When writing data to the card. On the CPSM Abort signal, hardware stops the
      IDMA and subsequently the FIFO is flushed.
   b) When reading data from the card. On the CPSM Abort signal, hardware instructs
      the IDMA to transfer the remaining data from the FIFO to RAM.
5. When the FIFO is empty/reset the DABORT flag is generated.

Stream operation and CMD12
To stop the stream transfer after the last byte to be transferred, the CMD12 end bit timing
must be sent aligned with the data stream end of last byte. The following write stream data
procedure applies:
1. Initialize the stream data in the DPSM, DTMODE = MCC stream data transfer.
2. Send the WRITE_DATA_STREAM command from the CPSM with CMDTRANS = 1.
3. Preload CMD12 in command registers, with the CMDSTOP bit set.
4. Configure the CPSM to send a command only after a wait pending (WAITPEND = 1)
   end of last data (according DATALENGTH).
5. Enabling the CPSM to send the STOP_TRANSMISSION command, the stream data
   end bit and command end bit are aligned.
   a) When DATALENGTH > 5 bytes, Command CMD12 is waited in the CPSM to be
      aligned with the data transfer end bit.
   b) When DATALENGTH < 5 bytes, Command CMD12 is started before and the
      DPSM remains in the Wait_S state to align the data transfer end with the CMD12
      end bit.
6. The write stream data can be aborted any time by clearing the WAITPEND bit. This
   causes the Preloaded CMD12 to be sent immediately and stop the write data stream.
To stop the read stream transfer after the last byte, the CMD12 end bit timing must occur after the last data stream byte. The following read stream data procedure applies:

1. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   a) The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.

2. Send CMD12 by the CPSM.
   a) CMD12 stops the card sending data.

*Note:* The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

**Block operation and CMD12**

To stop block transfer at the end of the data, the CMD12 end bit must be sent after the last block end bit.

When writing data to the card the CMD12 end bit must be sent after the write data block CRC token end bit. This requires the CMD12 sending to be tied to the data block transmission timing. To stop an Open-ended Multiple block write, the following procedure applies:

1. Before starting the data transfer, set DTMODE to "block data transfer ending with STOP_TRANSMISSION command".

2. Wait for all data to be sent by the DPSM and the CRC token to be received, (DATAEND flag).
   a) The DPSM does not send more data than indicated by DATALENGTH.

3. Send CMD12 by the CPSM.
   a) CMD12 sets the card to Idle mode.

When reading data from the card the CMD12 end bit must be sent earliest at the same time as the card read data block last data bit. This requires the CMD12 sending to be tied to the data block reception timing. The following stop Open-ended Multiple block read data block procedure applies:
1. Before starting the data transfer, set DTMODE to “block data transfer ending with
   STOP_TRANSMISSION command”.
2. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   a) The DPSM does not receive more data than indicated by DATALENGTH, even if
      the card is sending more data.
3. Send CMD12 with CMDSTOP bit set by the CPSM.
   a) CMD12 stops the Card sending more data and set the card to Idle mode. Any
      ongoing block transfer is aborted by the Card.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even
when the card continues sending data.

24.6.3 Sleep (CMD5)

The eMMC card may be switched between a Sleep state and a Standby state by CMD5. In
the Sleep state the power consumption of the card is minimized and the Vcc power supply
may be switched off.

The CMD5 (SLEEP) is used to initiate the state transition from Standby state to Sleep state.
The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Sleep
state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:
1. Enable interrupt on BUSYD0END.
2. Send CMD5 (SLEEP).
3. On BUSYD0END interrupt, card is in Sleep state
4. Vcc power supply can be switched off

The CMD5 (AWAKE) is used to initiate the state transition from Sleep state to Standby state.
The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The
Standby state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:
1. Switch on Vcc power supply and wait unit minimum operating level is reached.
2. Enable interrupt on BUSYD0END.
3. Send CMD5 (AWAKE).
4. On BUSYD0END interrupt card is in Standby state.

The Vcc power supply can be switched off only after the Sleep state has been reached. The
Vcc supply must be reinstalled before CMD5 (AWAKE) is sent.
24.6.4 Interrupt mode (Wait-IRQ)

The host and card enter and exit interrupt mode (Wait-IRQ) simultaneously. In interrupt mode there is no data transfer. The only message allowed is an interrupt service request response from the card or the host. For the interrupt mode to work correctly the SDMMC_CK frequency must be set in accordance with the achievable SDMMC_CMD data rate in Open Drain mode, which depend on the capacitive load and pull-up resistor. The CLKDIV must be set >1, and the SETCLKRX must select either the sdmmc_io_in_ck or SDMMC_CLKin source.

The host must ensure that the card is in Standby state before issuing the CMD40 (GO_IRQ_STATE). While waiting for an interrupt response the SDMMC_CK clock signal must be kept active.

A card in interrupt mode (IRQ state):
- is waiting for an internal card interrupt event. Once the event occurs, the card starts to send the interrupt service request response. The response is sent in open-drain mode.
- while waiting for the internal card interrupt event, the card also monitors the SDMMC_CMD line for a start bit. Upon detection of a start bit the card aborts the interrupt mode and switch to Standby state.

The host in interrupt mode (CPSM Wait state waiting for interrupt):
- is waiting for a card interrupt service request response (start bit).
- while waiting for a card interrupt service request response the host may abort the interrupt mode (by clearing the WAITINT register bit), which causes the host to send an interrupt service request response R5 with RCA = 0x0000 in open-drain mode.

When sending the interrupt service request response, the sender bit-wise monitors the SDMMC_CMD bit stream. The sender whose interrupt service request response bit does not correspond to the bit on the SDMMC_CMD line stops sending. In the case of multiple senders only one successfully sends its full interrupt service request response. If the host sends simultaneously, it loses sending after the transmission bit.

To handle the interrupt mode, the following procedure applies:

![CMD5 Sleep Awake procedure](image)
1. Set the SDMMC_CK frequency in accordance with the achievable SDMMC_CMD data rate in Open-drain mode, CLKDIV must be set >1, and SETCLKRX must select the sdmmc_io_in_ck.

2. Load CMD40 (GO_IRQ_STATE) in the command registers.

3. Enable wait for interrupt by setting WAITINT register bit.

4. Configure the CPSM to send a command immediately.
   a) This causes the CMD40 to be sent and the CPSM to be halted in the Wait state, waiting for an interrupt service request response.

5. To exit the wait for interrupt state (CPSM Wait state):
   a) Upon the detection of an interrupt service request response start bit the CPSM moves to the Receive state where the response is received. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.
   b) To abort the interrupt mode the host clears the WAITINT register bit, which causes the host to send an interrupt service request response by itself. This moves the CPSM to the Receive state. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.

*Note:* On a simultaneous send interrupt service request response start bit collision the host loses the bus access after the transmission bit.

### 24.6.5 Boot operation

In boot operation mode the host can read boot data from the card by either one of the 2 boot operation functions:

1. Normal boot. (keeping CMD line low)
2. Alternative boot (sending CMD0 with argument 0xFFFFFFFFA)

The boot data can be read according to the following configuration options, depending on card register settings:

- The partition from which boot data is read (EXT_CSD Byte[179])
- The boot data size (EXT_CSD Byte[226])
- The bus configuration during boot (EXT_CSD Byte[177])
- Receiving boot acknowledgment from the card. (EXT_CSD Byte[179])

If boot acknowledgment is enabled the card send pattern 010 on SDMMC_D0 within 50ms after boot mode has been requested by either CMD line going low or after CMD0 with argument 0xFFFFFFFFA. A boot acknowledgment timeout (ACKTIMEOUT) and acknowledgment status (ACKFAIL) is provided.

**Normal boot operation**

If the SDMMC_CMD line is held low for at least 74 clock cycles after card power-up or reset, before the first command is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD line goes low, the card starts to send the first boot code data on the SDMMC_Dn line(s). The host must keep the SDMMC_CMD line low until after all boot data has been read. The host can terminate boot mode by pulling the SDMMC_CMD line high.
To perform the normal boot procedure the following steps are needed:

1. Reset the card.
2. If a boot acknowledgment is requested, enable the BOOTACKEN and set the ACKTIME and ACKFAIL and ACKTIMEOUT interrupt.
3. Enable data reception by setting the DPSM in receive mode (DTDIR) and the number of data bytes to be received in DATALENGTH.
4. Enable the DTIMEOUT, DATAEND, and CMDSENT interrupts for end of boot command confirmation.
5. Select the normal boot operation mode in BOOTMODE, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This causes:
   - the SDMMC_CMD to be driven low. (BOOTMODE = normal boot).
   - the ACK timeout to start.
   - DPSM to be enabled.
6. The incorrect reception of the boot acknowledgment can be detected with ACKFAIL flag or ACKTIMEOUT flag when enabled.
   - when an incorrect boot acknowledgment is received, the ACKFAIL flag occurs.
   - when the boot acknowledgment is not received in time, the ACKTIMEOUT flag occurs.
7. When all boot data has been received, the DATAEND flag occurs.
   - when data CRC fails, the DCRCFAIL flag is also generated.
   - when the data timeout occurs, the DTIMEOUT flag is also generated.
8. When the last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   - SDMMC has completely received all data and the DPSM is disabled.
9. The boot procedure is terminated by firmware clearing BOOTEN, which causes the SDMMC_CMD line to go high. The CMDSENT flag is generated 56 cycles later to indicate that a new command can be sent.
   a) If the boot procedure is aborted by firmware before all data has been received, the CPSM Abort signal stops data reception and disables the DPSM which triggers an DABORT flag when enabled.
10. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command.
Alternative boot operation

After card power-up or reset, if the host send CMD0 with the argument 0xFFFFFFFFFA after 74 clock cycles before CMD0 is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD0 with argument 0xFFFFFFFFFA has been sent, the card starts to send the first boot code data on the SDMMC_Dn line(s). The master terminates boot operation by sending CMD0 (Reset).

**Figure 180. Alternative boot mode operation**

To perform the alternative boot procedure the following steps needed:

1. Move the SDMMC to power-off state, and reset the card
2. Move the SDMMC to power-on state. This guarantees the 74 SCDMMC_CK cycles to be clocked before any command.
3. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKTIMEOUT flag.
4. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data to be received in DATALENGTH. Enable the DTIMEOUT and DATAEND flags.
5. Select the alternative boot operation mode in BOOTMODE, load the CMD0 with the 0xFFFFFFFFFA argument in the command registers. Enable CMDSENT flag for end of
boot command confirmation, and enable boot in BOOTEN. The boot procedure is 
started by enabling the CPSM with CPSMEN. This causes:
   – the loaded command and argument to be sent out. (BOOTMODE = alternative 
   boot).
   – the ACK timeout to start.
   – DPSM to be enabled.
6. When the command has been sent the CMDSENT flag is generated, at which time the 
BOOTEN bit must be cleared.
7. the reception of the boot acknowledgment can be detected with ACKFAIL flag when 
enabled.
   – when the boot acknowledgment is not received in time the ACKTIMEOUT flag 
occurs.
8. when all boot data has been received the DATAEND flag occurs.
   – when data CRC fails the DCRCFAIL flag is also generated.
   – when the data timeout occurs the DTIMEOUT flag is also generated.
9. When last data has been received, read data from the FIFO until FIFO is empty after 
which end of data DATAEND flag is generated.
   – SDMMC has completely received all data and the DPSM is disabled.
10. The BOOTEN bit must be cleared, before terminating the boot procedure by sending 
CMD0 (Reset) with BOOTMODE = alternative boot. This causes the CMDSENT flag to 
occur 56 cycles after the Command.
   – if the boot procedure is aborted by firmware before all data has been received the 
   CPSM Abort signal stops the data transfer and disable the DPSM which triggers 
an DABORT flag when enabled.
11. The CMDSENT flag signals the end of the boot procedure and the card is ready to 
receive a new command. When the RESET command has been sent successfully, the 
BOOTMODE control bit has to be cleared to terminate the boot operation.

24.6.6 Response R1b handling

When sending commands which have a R1b response the busy signaling is reflected in the 
BUSYD0 register bit and the release of busy with the BUSYD0END flag. The SDMMC_D0 
line is sampled at the end of the R1b response and signaled in the BUSYD0 register bit. The 
BUSYD0 register bit is reset to not busy when the SDMMC_D0 line release busy, at the 
same time the BUSYD0END flag is generated.
The expected maximum busy time must be set in the DATATIME register before sending the command. When enabled, the DTIMEOUT flag is set when after the R1b response busy stays active longer then the programmed time.

To detect the SDMMC_D0 busy signaling when sending a Command with R1b response the following procedure applies:

- Enable CMDREND flag
- Send Command through CPSM.
- On the CMDREND flag check the BUSYD0 register bit.
  - If BUSYD0 signals not busy, signal busy release to the Firmware
  - If BUSYD0 signals busy, wait for BUSYD0END flag
- On BUSYD0END flag signal busy released to the firmware.
- On DTIMEOUT flag busy is active longer then programmed time.

### 24.6.7 Reset and card cycle power

**Reset**

Following reset the SDMMC is in the reset state. In this state the SDMMC is disabled and no command nor data can be transfered. The SDMMC_D[7:0], and SDMMC_CMD are in HiZ and the SDMMC_CK is driven low.

Before moving to the power-on state the SDMMC must be configured.

In the power-on state the SDMMC_CK clock is running. First 74 SDMMC_CK cycles are clocked after which the SDMMC is enabled and command and data can be transfered.

The SDMMC states are controlled by Firmware with the PWRCTRL register bits according to Figure 182.

**Card cycle power**

To perform a card cycle power the following procedure applies:
1. Reset the SDMMC with the RCC.SDMMCxRST register bit. This resets the SDMMC to the reset state and the CPSM and DPSM to the Idle state.

2. Disable the Vcc power to the card.

3. Set the SDMMC in power-cycle state. This makes that the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low, to prevent the card from being supplied through the signal lines.

4. After minimum 1 ms enable the Vcc power to the card.

5. After the power ramp period set the SDMMC to the power-off state for minimum 1 ms. The SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are set to drive “1”.

6. After the 1 ms delay set the SDMMC to power-on state in which the SDMMC_CK clock is enabled.

7. After 74 SDMMC_CK cycles the first command can be sent to the card.

Figure 183. Card cycle power / power up diagram

24.7 Hardware flow control

The hardware flow control during data transfer functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

The behavior is to stop SDMMC_CK during data transfer and freeze the SDMMC state machines. The data transfer is stalled when the FIFO is unable to transmit or receive data. The data transfer remains stalled until the transmit FIFO is half full or all data according DATALENIGHT has been stored, or until the receive FIFO is half empty. Only state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be filled or emptied even if flow control is activated.

On an IDMA linked list transfer error, the hardware flow control is disabled. As a consequence, depending on when the IDMA linked list transfer error occurs, an underrun or overrun error may also occur (see Section : IDMA transfer error management).
To enable hardware flow control during data transfer, the HWFC_EN register bit must be set to 1. After reset hardware flow control is disabled.

Hardware flow control must only be used when the SDMMC_Dn data is cycle-aligned with the SDMMC_CK. Whenever the sdmmc_fb_ck from the DLYB delay block is used, i.e. in the case of SDR104 mode with a t\text{OP} and Dt\text{OP} delay > 1 cycle, hardware flow control can not be used.

### 24.8 Ultra-high-speed phase I (UHS-I) voltage switch

UHS-I mode (SDR12, SDR25, SDR50, SDR104, and DDR50) requires the support for 1.8V signaling. After power up the card starts in 3.3V mode. CMD11 invokes the voltage switch sequence to the 1.8V mode. When the voltage sequence is completed successfully the card enters UHS-I mode with default SDR12 and card input and output timings are changed.

**Figure 184. CMD11 signal voltage switch sequence**

To perform the signal voltage switch sequence the following steps are needed:

1. Before starting the Voltage Switch procedure, the SDMMC_CK frequency must be set in the range 100 kHz - 400 kHz.
2. The host starts the Voltage Switch procedure by setting the VSWITCHEN bit before sending the CMD11.
3. The card returns an R1 response.
   - if the response CRC is pass, the Voltage Switch procedure continues the host does no longer drive the CMD and SDMMC_D[3:0] signals until completion of the voltage switch sequence. Some cycles after the response the SDMMC_CK is stopped and the CKSTOP flag is set.
   - if the response CRC is fail (CCRCFAIL flag) or no response is received before the timeout (CTIMEOUT flag), the Voltage Switch procedure is stopped.
4. The card drives CMD and SDMMC_D[3:0] to low at the next clock after the R1 response.
5. The host, after having received the R1 response, may monitor the SDMMC_D0 line using the BUSYD0 register bit. The SDMMC_D0 line is sampled two SDMMC_CK clock cycles after the Response. The Firmware may read the BUSYD0 register bit following the CKSTOP flag.
   - When the BUSYD0 is detected low the host firmware switches the Voltage regulator to 1.8V, after which it instructs the SDMMC to start the timing critical
section of the Voltage Switch sequence by setting register bit VSWITCH. The hardware continues to stop the SDMMC_CK by holding it low for at least 5 ms.

– When the BUSYD0 is detected high the host aborts the Voltage Switch sequence and cycle power the card.

6. The card after detecting SDMMC_CK low begins switching signaling voltage to 1.8 V.

7. The host SDMMC hardware after at least 5 ms restarts the SDMMC_CK.

8. The card within 1 ms from detecting SDMMC_CK transition drives CMD and DAT[3:0] high for at least 1 SDMMC_CK cycle and then stop driving CMD and DAT[3:0].

9. The host SDMMC hardware, 1 ms after the SDMMC_CK has been restarted, the SDMMC_D0 is sampled into BUSYD0 and the VSWEND flag is set.

10. The host, on the VSWEND flag, checks SDMMC_D0 line using the BUSYD0 register bit, to confirm completion of voltage switch sequence:

– When BUSYD0 is detected high, Voltage Switch has been completed successfully.
– When BUSYD0 is detected low, Voltage Switch has failed, the host cycles the card power.

The minimum 5 ms time to stop the SDMMC_CK is derived from the internal un-gated SDMMC_CK clock, which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The >5 ms time is counted by $2^{12}$ cycles (10.24 ms @ 400 kHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time for the SDMMC_CK clock to be stopped is longer.

The maximum 1 ms time for the card to drive the SDMMC_Dn and SDMMC_CMD lines high is derived from the internal ungated SDMMC_CK which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The SDMMC checks the lines after >1 ms time which is counted by $2^5$ cycles (1.28 ms @ 25 MHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time to check the lines is longer.

The signal voltage level is supported through an external voltage translation transceiver like STMicroelectronics ST6G3244ME.
To interface with an external driver (a voltage switch transceiver), next to the standard signals the SDMMC uses the following signals:

**SDMMC_CKIN** feedback input clock

**SDMMC_CDIR** I/O direction control for the CMD signal.

**SDMMC_D0DIR** I/O direction control for the SDMMC_D0 signal.

**SDMMC_D123DIR** I/O direction control for the SDMMC_D1, SDMMC_D2 and SDMMC_D3 signals.

The voltage transceiver signals **EN** and **SEL** are to be handled through general-purpose I/O.

The polarity of the SDMMC_CDIR, SDMMC_D0DIR and SDMMC_D123DIR signals can be selected through SDMMC_POWER.DIRPOL control bit.
## 24.9 SDMMC interrupts

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMMC</td>
<td>Command response CRC fail</td>
<td>CCRCFAIL</td>
<td>CCRCFAILIE</td>
<td>CCRCFAILC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data block CRC fail</td>
<td>DCRCFAIL</td>
<td>DCRCFAILIE</td>
<td>DCRCFAILC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Command response timeout</td>
<td>CTIMEOUT</td>
<td>CTIMEOUTIE</td>
<td>CTIMEOUTC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data timeout</td>
<td>DTIMEOUT</td>
<td>DTIMEOUTIE</td>
<td>DTIMEOUTC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Transmit FIFO underrun</td>
<td>TXUNDERR</td>
<td>TXUNDERRIE</td>
<td>TXUNDERRC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Receive FIFO overrun</td>
<td>RXOVERR</td>
<td>RXOVERRIE</td>
<td>RXOVERRC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Command response received</td>
<td>CMDREND</td>
<td>CMDRENDIE</td>
<td>CMDRENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Command sent</td>
<td>CMDSENT</td>
<td>CMDSENTIE</td>
<td>CMDSENTC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data transfer ended</td>
<td>DATAEND</td>
<td>DATAENDIE</td>
<td>DATAENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data transfer hold</td>
<td>DHOLD</td>
<td>DHOLDIE</td>
<td>DHOLDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data block sent or received</td>
<td>DBCKEND</td>
<td>DBCKENDIE</td>
<td>DBCKENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data transfer aborted</td>
<td>DABORT</td>
<td>DABORTIE</td>
<td>DABORTC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Transmit FIFO half empty</td>
<td>TXFIFOHE</td>
<td>TXFIFOHEIE</td>
<td>n.a.</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Receive FIFO half full</td>
<td>RXFIFOHF</td>
<td>RXFIFOHFIE</td>
<td>n.a.</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Transmit FIFO full</td>
<td>TXFIFO</td>
<td>n.a.</td>
<td>n.a.</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Receive FIFO full</td>
<td>RXFIFO</td>
<td>RXFIFOIE</td>
<td>n.a.</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Transmit FIFO empty</td>
<td>TXFIFOE</td>
<td>TXFIFOEIE</td>
<td>n.a.</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Receive FIFO empty</td>
<td>RXFIFOE</td>
<td>n.a.</td>
<td>n.a.</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Table 233. SDMMC interrupts (continued)

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMMC</td>
<td>Command response end of busy</td>
<td>BUSYD0END</td>
<td>BUSYD0ENDIE</td>
<td>BUSYD0ENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>SDIO interrupt</td>
<td>SDIOIT</td>
<td>SDIOITIE</td>
<td>SDIOITC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Boot acknowledgment fail</td>
<td>ACKFAIL</td>
<td>ACKFAILIE</td>
<td>ACKFAILC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Boot acknowledgment timeout</td>
<td>ACKTIMEOUT</td>
<td>ACKTIMEOUTIE</td>
<td>ACKTIMEOUTC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Voltage switch timing</td>
<td>VSWEND</td>
<td>VSWENDIE</td>
<td>VSWENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>SDMM_CK stopped in voltage switch</td>
<td>CKSTOP</td>
<td>CKSTOPIE</td>
<td>CKSTOPOC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>IDMA transfer error</td>
<td>IDMATE</td>
<td>IDMATEIE</td>
<td>IDMATEC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>IDMA buffer transfer complete</td>
<td>IDMABTC</td>
<td>IDMABTCIE</td>
<td>IDMABTCC</td>
<td>Yes</td>
</tr>
</tbody>
</table>
24.10 SDMMC registers

The device communicates to the system via 32-bit control registers accessible via AHB slave interface.

The peripheral registers have to be accessed by words (32-bit). Byte (8-bit) and halfword (16-bit) accesses trigger an AHB bus error.

24.10.1 SDMMC power control register (SDMMC_POWER)

Address offset: 0x000

Reset value: 0x0000 0000

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

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

Bit 4 DIRPOL: Data and command direction signals polarity selection
This bit can only be written when the SDMMC is in the power-off state (PWRCTRL = 00).
0: Voltage transceiver IOs driven as output when direction signal is low.
1: Voltage transceiver IOs driven as output when direction signal is high.

Bit 3 VSWITCHEN: Voltage switch procedure enable
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
This bit is used to stop the SDMMC_CK after the voltage switch command response:
0: SDMMC_CK clock kept unchanged after successfully received command response.
1: SDMMC_CK clock stopped after successfully received command response.

Bit 2 VSWITCH: Voltage switch sequence start
This bit is used to start the timing critical section of the voltage switch sequence:
0: Voltage switch sequence not started and not active.
1: Voltage switch sequence started or active.

Bits 1:0 PWRCTRL[1:0]: SDMMC state control bits
These bits can only be written when the SDMMC is not in the power-on state (PWRCTRL ≠ 11).
These bits are used to define the functional state of the SDMMC signals:
00: After reset, Reset: the SDMMC is disabled and the clock to the Card is stopped, SDMMC_D[7:0], and SDMMC_CMD are HiZ and SDMMC_CK is driven low.
When written 00, power-off: the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven high.
01: Reserved. (When written 01, PWRCTRL value does not change)
10: Power-cycle, the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low.
11: Power-on: the card is clocked, The first 74 SDMMC_CK cycles the SDMMC is still disabled. After the 74 cycles the SDMMC is enabled and the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are controlled according the SDMMC operation.
Any further write is ignored, PWRCTRL value keeps 11.
24.10.2 SDMMC clock control register (SDMMC_CLKCR)

Address offset: 0x004
Reset value: 0x0000 0000

The SDMMC_CLKCR register controls the SDMMC_CK output clock, the sdmmc_rx_ck receive clock, and the bus width.

| Bits 31:22 | Reserved, must be kept at reset value. |
| Bits 21:20 | **SELCLKRX[1:0]**: Receive clock selection |
| These bits can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0) |
| 00: sdmmc_io_in_ck selected as receive clock |
| 01: SDMMC_CKIN feedback clock selected as receive clock |
| 10: sdmmc_fb_ck tuned feedback clock selected as receive clock. |
| 11: Reserved (select sdmmc_io_in_ck) |

| Bit 19 | **BUSSPEED**: Bus speed for selection of SDMMC operating modes |
| This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0) |
| 0: DS, HS, SDR12, SDR25, Legacy compatible, High speed SDR, High speed DDR bus speed mode selected |
| 1: SDR50, DDR50, SDR104, HS200 bus speed mode selected. |

| Bit 18 | **DDR**: Data rate signaling selection |
| This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0) |
| DDR rate must only be selected with 4-bit or 8-bit wide bus mode. (WIDBUS > 00). DDR = 1 has no effect when WIDBUS = 00 (1-bit wide bus). |
| If DDR rate must only be selected with clock division >1. (CLKDIV > 0) |
| 0: SDR Single data rate signaling |
| 1: DDR double data rate signaling |

| Bit 17 | **HWFC_EN**: Hardware flow control enable |
| This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0) |
| 0: Hardware flow control is disabled |
| 1: Hardware flow control is enabled |
| When Hardware flow control is enabled, the meaning of the TXFIFOE and RXFIFOF flags change, see SDMMC status register definition in Section 24.10.11. |
Bit 16 **NEGEDGE**: SDMMC\_CK dephasing selection bit for data and command
This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).
When clock division = 1 (CLKDIV = 0), this bit has no effect. Data and Command change on SDMMC\_CK falling edge.
0: When clock division >1 (CLKDIV > 0) and DDR = 0:
– Command and data changed on the sdmcc\_ker\_ck falling edge succeeding the rising edge of SDMMC\_CK.
– SDMMC\_CK edge occurs on sdmcc\_ker\_ck rising edge.
When clock division >1 (CLKDIV > 0) and DDR = 1:
– Command changed on the sdmcc\_ker\_ck falling edge succeeding the rising edge of SDMMC\_CK.
– Data changed on the sdmcc\_ker\_ck falling edge succeeding a SDMMC\_CK edge.
– SDMMC\_CK edge occurs on sdmcc\_ker\_ck rising edge.
1: When clock division >1 (CLKDIV > 0) and DDR = 0:
– Command and data changed on the same sdmcc\_ker\_ck rising edge generating the SDMMC\_CK falling edge.
When clock division >1 (CLKDIV > 0) and DDR = 1:
– Command changed on the same sdmcc\_ker\_ck rising edge generating the SDMMC\_CK falling edge.
– Data changed on the SDMMC\_CK falling edge succeeding a SDMMC\_CK edge.
– SDMMC\_CK edge occurs on sdmcc\_ker\_ck rising edge.

Bits 15:14 **WIDBUS[1:0]**: Wide bus mode enable bit
This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).
00: Default 1-bit wide bus mode: SDMMC\_D0 used (Does not support DDR)
01: 4-bit wide bus mode: SDMMC\_D[3:0] used
10: 8-bit wide bus mode: SDMMC\_D[7:0] used

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PWRSAV**: Power saving configuration bit
This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).
For power saving, the SDMMC\_CK clock output can be disabled when the bus is idle by setting PWRSAV:
0: SDMMC\_CK clock is always enabled
1: SDMMC\_CK is only enabled when the bus is active

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **CLKDIV[9:0]**: Clock divide factor
This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).
This field defines the divide factor between the input clock (sdmmc\_ker\_ck) and the output clock (SDMMC\_CK):
SDMMC\_CK frequency = sdmmc\_ker\_ck / [2 * CLKDIV].
0x000: SDMMC\_CK frequency = sdmmc\_ker\_ck / 1 (Does not support DDR)
0x001: SDMMC\_CK frequency = sdmmc\_ker\_ck / 2
0x002: SDMMC\_CK frequency = sdmmc\_ker\_ck / 4
0x0XX: etc..
0x080: SDMMC\_CK frequency = sdmmc\_ker\_ck / 256
0xXXX: etc..
0x3FF: SDMMC\_CK frequency = sdmmc\_ker\_ck / 2046
Note: 1 While the SD/SDIO card or e-MMC is in identification mode, the SDMMC_CK frequency must be less than 400 kHz.

2 The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

3 At least seven sdmmc_hclk clock periods are needed between two write accesses to this register. SDMMC_CK can also be stopped during the Read Wait interval for SD I/O cards: in this case the SDMMC_CLKCR register does not control SDMMC_CK.

### 24.10.3 SDMMC argument register (SDMMC_ARGR)

Address offset: 0x008

Reset value: 0x0000 0000

The SDMMC_ARGR register contains a 32-bit command argument, which is sent to a card as part of a command message.

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

Bits 31:0 CMDARG[31:0]: Command argument

These bits can only be written by firmware when CPSM is disabled (CPSMEN = 0). Command argument sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

### 24.10.4 SDMMC command register (SDMMC_CMDR)

Address offset: 0x00C

Reset value: 0x0000 0000

The SDMMC_CMDR register contains the command index and command type bits. The command index is sent to a card as part of a command message. The command type bits control the command path state machine (CPSM).

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

<table>
<thead>
<tr>
<th>BOOT EN</th>
<th>BOOT MODE</th>
<th>DT HOLD</th>
<th>CPSM EN</th>
<th>WAITP END</th>
<th>WAIT INT</th>
<th>WAITRESP[1:0]</th>
<th>CMD STOP</th>
<th>CMD TRANS</th>
<th>CMDINDEX[5:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:17  Reserved, must be kept at reset value.

Bit 16 **CMDSUSPEND**: The CPSM treats the command as a Suspend or Resume command and signals interrupt period start/end
   This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
   CMDSUSPEND = 1 and CMDTRANS = 0 Suspend command, start interrupt period when response bit BS=0.
   CMDSUSPEND = 1 and CMDTRANS = 1 Resume command with data, end interrupt period when response bit DF=1.

Bit 15 **BOOTEN**: Enable boot mode procedure
   0: Boot mode procedure disabled
   1: Boot mode procedure enabled

Bit 14 **BOOTMODE**: Select the boot mode procedure to be used
   This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0)
   0: Normal boot mode procedure selected
   1: Alternative boot mode procedure selected.

Bit 13 **DTHOLD**: Hold new data block transmission and reception in the DPSM
   If this bit is set, the DPSM does not move from the Wait_S state to the Send state or from the Wait_R state to the Receive state.

Bit 12 **CPSMEN**: Command path state machine (CPSM) enable bit
   This bit is written 1 by firmware, and cleared by hardware when the CPSM enters the Idle state.
   If this bit is set, the CPSM is enabled.
   When DTEN = 1, no command is transferred nor boot procedure is started. CPSMEN is cleared to 0.
   During Read Wait with SDMMC_CK stopped no command is sent and CPSMEN is kept 0.

Bit 11 **WAITPEND**: CPSM waits for end of data transfer (CmdPend internal signal) from DPSM
   This bit when set, the CPSM waits for the end of data transfer trigger before it starts sending a command.
   WAITPEND is only taken into account when DTMODE = eMMC stream data transfer,
   WIDBUS = 1-bit wide bus mode, DPSMACT = 1 and DTDIR = from host to card.

Bit 10 **WAITINT**: CPSM waits for interrupt request
   If this bit is set, the CPSM disables command timeout and waits for an card interrupt request (Response).
   If this bit is cleared in the CPSM Wait state, it causes the abort of the interrupt mode.

Bits 9:8 **WAITRESP[1:0]**: Wait for response bits
   This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
   They are used to configure whether the CPSM is to wait for a response, and if yes, which kind of response.
   00: No response, expect CMDSENT flag
   01: Short response, expect CMDREND or CCRCFAIL flag
   10: Short response, expect CMDREND flag (No CRC)
   11: Long response, expect CMDREND or CCRCFAIL flag
Bit 7 **CMDSTOP**: The CPSM treats the command as a Stop Transmission command and signals abort to the DPSM.

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues the abort signal to the DPSM when the command is sent.

Bit 6 **CMDTRANS**: The CPSM treats the command as a data transfer command, stops the interrupt period, and signals DataEnable to the DPSM.

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues an end of interrupt period and issues DataEnable signal to the DPSM when the command is sent.

Bits 5:0 **CMDINDEX[5:0]**: Command index

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

The command index is sent to the card as part of a command message.

---

**Note:**

1. At least seven sdmmc_hclk clock periods are needed between two write accesses to this register.

2. MultiMediaCard can send two kinds of response: short responses, 48 bits, or long responses, 136 bits. SD card and SD I/O card can send only short responses, the argument can vary according to the type of response: the software distinguishes the type of response according to the send command.

---

**24.10.5 SDMMC command response register (SDMMC_RESPCMDR)**

Address offset: 0x010

Reset value: 0x0000 0000

The SDMMC_RESPCMDR register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long or OCR response), the RESPCMD field is unknown, although it must contain 111111b (the value of the reserved field from the response).

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

Bits 5:0 **RESPCMD[5:0]**: Response command index

Read-only bit field. Contains the command index of the last command response received.
24.10.6 SDMMC response x register (SDMMC_RESPxR)

Address offset: 0x010 + 0x004 * x, (x = 1 to 4)

Reset value: 0x0000 0000

The SDMMC_RESP1/2/3/4R registers contain the status of a card, which is part of the received response.

<table>
<thead>
<tr>
<th>CARDSTATUS[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CARDSTATUS[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r r r r</td>
</tr>
</tbody>
</table>

Bits 31:0 CARDSTATUS[31:0]: Card status according to the table below.

See Table 234.

The card status size is 32 or 128 bits, depending on the response type.

Table 234. Response type and SDMMC_RESPxR registers

<table>
<thead>
<tr>
<th>Register</th>
<th>Short response</th>
<th>Long response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMMC_RESP1R</td>
<td>Card status[31:0]</td>
<td>Card status[127:96]</td>
</tr>
<tr>
<td>SDMMC_RESP2R</td>
<td>all 0</td>
<td>Card status[95:64]</td>
</tr>
<tr>
<td>SDMMC_RESP3R</td>
<td>all 0</td>
<td>Card status[63:32]</td>
</tr>
<tr>
<td>SDMMC_RESP4R</td>
<td>all 0</td>
<td>Card status[31:0][2]</td>
</tr>
</tbody>
</table>

1. The most significant bit of the card status is received first.

2. The SDMMC_RESP4R register LSB is always 0.

24.10.7 SDMMC data timer register (SDMMC_DTIMER)

Address offset: 0x024

Reset value: 0x0000 0000

The SDMMC_DTIMER register contains the data timeout period, in card bus clock periods.

A counter loads the value from the SDMMC_DTIMER register, and starts decrementing when the data path state machine (DPSM) enters the Wait_R or Busy state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

<table>
<thead>
<tr>
<th>DATATIME[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATATIME[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>
Bits 31:0  DATATIME[31:0]: Data and R1b busy timeout period
   This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and
   DPSMACT = 0).
   Data and R1b busy timeout period expressed in card bus clock periods.

Note:  A data transfer must be written to the data timer register and the data length register before
being written to the data control register.

24.10.8  SDMMC data length register (SDMMC_DLENR)

Address offset: 0x028

Reset value: 0x0000 0000

The SDMMC_DLENR register contains the number of data bytes to be transferred. The
value is loaded into the data counter when data transfer starts.

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

Bits 31:25  Reserved, must be kept at reset value.

Bits 24:0  DATALENGTH[24:0]: Data length value
   This register can only be written by firmware when DPSM is inactive (DPSMACT = 0).
   Number of data bytes to be transferred.
   When DDR = 1 DATALENGTH is truncated to a multiple of 2. (The last odd byte is not
   transferred)
   When DATALENGTH = 0 no data are transferred, when requested by a CPSMEN and
   CMDTRANS = 1 also no command is transfered. DTEN and CPSMEN are cleared to 0.

Note:  For a block data transfer, the value in the data length register must be a multiple of the block
size (see SDMMC_DCTRL). A data transfer must be written to the data timer register and
the data length register before being written to the data control register.

For an SDMMC multibyte transfer the value in the data length register must be between 1
and 512.
24.10.9  SDMMC data control register (SDMMC_DCTRL)

Address offset: 0x02C
Reset value: 0x0000 0000

The SDMMC_DCTRL register control the data path state machine (DPSM).

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-14</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>13</td>
<td>FIFORST: FIFO reset, flushes any remaining data</td>
</tr>
<tr>
<td>12</td>
<td>BOOTACKEN: Enable the reception of the boot acknowledgment</td>
</tr>
<tr>
<td>11</td>
<td>SDIOEN: SD I/O interrupt enable functions</td>
</tr>
<tr>
<td>10</td>
<td>RWMOD: Read Wait mode</td>
</tr>
<tr>
<td>9</td>
<td>RWSTOP: Read Wait stop</td>
</tr>
<tr>
<td>8</td>
<td>RWSTART: Read Wait start</td>
</tr>
</tbody>
</table>

Bits 31:14  Reserved, must be kept at reset value.

Bit 13  **FIFORST**: FIFO reset, flushes any remaining data
- This bit can only be written by firmware when IDMAEN= 0 and DPSM is active (DPSMACT = 1). This bit only takes effect when a transfer error or transfer hold occurs.
- 0: FIFO not affected.
- 1: Flush any remaining data and reset the FIFO pointers. This bit is automatically cleared to 0 by hardware when DPSM gets inactive (DPSMACT = 0).

Bit 12  **BOOTACKEN**: Enable the reception of the boot acknowledgment
- This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
- 0: Boot acknowledgment disabled, not expected to be received
- 1: Boot acknowledgment enabled, expected to be received

Bit 11  **SDIOEN**: SD I/O interrupt enable functions
- This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
- If this bit is set, the DPSM enables the SD I/O card specific interrupt operation.

Bit 10  **RWMOD**: Read Wait mode
- This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
- 0: Read Wait control using SDMMC_D2
- 1: Read Wait control stopping SDMMC_CK

Bit 9  **RWSTOP**: Read Wait stop
- This bit is written by firmware and auto cleared by hardware when the DPSM moves from the R_W state to the Wait_R or Idle state.
- 0: No Read Wait stop.
- 1: Enable for Read Wait stop when DPSM is in the R_W state.

Bit 8  **RWSTART**: Read Wait start
- If this bit is set, Read Wait operation starts.
Bits 7:4  **DBLOCKSIZE[3:0]:** Data block size

This bit can only be written by firmware when DPSM is inactive (DPSM_ACT = 0).
Define the data block length when the block data transfer mode is selected:

- 0000: Block length = $2^0 = 1$ byte
- 0001: Block length = $2^1 = 2$ bytes
- 0010: Block length = $2^2 = 4$ bytes
- 0011: Block length = $2^3 = 8$ bytes
- 0100: Block length = $2^4 = 16$ bytes
- 0101: Block length = $2^5 = 32$ bytes
- 0110: Block length = $2^6 = 64$ bytes
- 0111: Block length = $2^7 = 128$ bytes
- 1000: Block length = $2^8 = 256$ bytes
- 1001: Block length = $2^9 = 512$ bytes
- 1010: Block length = $2^{10} = 1024$ bytes
- 1011: Block length = $2^{11} = 2048$ bytes
- 1100: Block length = $2^{12} = 4096$ bytes
- 1101: Block length = $2^{13} = 8192$ bytes
- 1110: Block length = $2^{14} = 16384$ bytes
- 1111: Reserved

When DATALENGTH is not a multiple of DBLOCKSIZE, the transferred data is truncated at a multiple of DBLOCKSIZE. (None of the remaining data are transferred.)

When DDR = 1, DBLOCKSIZE = 0000 must not be used. (No data are transferred)

Bits 3:2  **DTMODE[1:0]:** Data transfer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSM_ACT = 0).

- 00: Block data transfer ending on block count.
- 01: SDIO multibyte data transfer.
- 10: e-MMC Stream data transfer. (WIDBUS must select 1-bit wide bus mode)
- 11: Block data transfer ending with STOP_TRANSMISSION command (not to be used with DTEN initiated data transfers).

Bit 1  **DTDIR:** Data transfer direction selection

This bit can only be written by firmware when DPSM is inactive (DPSM_ACT = 0).

- 0: From host to card.
- 1: From card to host.

Bit 0  **DTEN:** Data transfer enable bit

This bit can only be written by firmware when DPSM is inactive (DPSM_ACT = 0). This bit is cleared by Hardware when data transfer completes.

This bit must only be used to transfer data when no associated data transfer command is used, i.e. must not be used with SD or e-MMC cards.

- 0: Do not start data transfer without CPSM data transfer command.
- 1: Start data transfer without CPSM data transfer command.

### 24.10.10 SDMMC data counter register (SDMMC_DCNTR)

Address offset: 0x030

Reset value: 0x0000 0000

The SDMMC_DCNTR register loads the value from the data length register (see SDMMC_DLENR) when the DPSM moves from the Idle state to the Wait_R or Wait_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then
moves to the Idle state and when there has been no error, and no transmit data transfer hold, the data status end flag (DATAEND) is set.

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

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATACOUNT[24:0]**: Data count value
When read, the number of remaining data bytes to be transferred is returned. Write has no effect.

**Note:** This register must be read only after the data transfer is complete, or hold. When reading after an error event the read data count value may be different from the real number of data bytes transferred.

### 24.10.11 SDMMC status register (SDMMC_STAR)

Address offset: 0x034
Reset value: 0x0000 0000

The SDMMC_STAR register is a read-only register. It contains two types of flag:
- Static flags (bits [28, 21, 11:0]): these bits remain asserted until they are cleared by writing to the SDMMC interrupt Clear register (see SDMMC_ICR)
- Dynamic flags (bits [20:12]): these bits change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>IDMABTC</td>
<td>IDMA TE</td>
<td>CK STOP</td>
<td>VSW END</td>
<td>ACK TIME OUT</td>
<td>ACK FAIL</td>
<td>SDIOIT</td>
<td>BUSY D0END</td>
<td>BUSY D1</td>
<td>RX FIFOE</td>
<td>TX FIFOE</td>
<td>RX FIFOF</td>
<td>TX FIFOF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RX FIFOHF</td>
<td>TX FIFOHE</td>
<td>CPSM ACT</td>
<td>DPSM ACT</td>
<td>DA BORT</td>
<td>DBCK END</td>
<td>DHOLD</td>
<td>DATA END</td>
<td>CMD SENT</td>
<td>CMDR END</td>
<td>RX OVERR</td>
<td>TX UNDER R</td>
<td>D TIME OUT</td>
<td>C TIME OUT</td>
<td>DCR FAIL</td>
<td>CCRC FAIL</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTC**: IDMA buffer transfer complete
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 27 **IDMATE**: IDMA transfer error
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 26 **CKSTOP**: SDMMC_CK stopped in Voltage switch procedure
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
Bit 25 **VSWEND**: Voltage switch critical timing section completion  
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 24 **ACKTIMEOUT**: Boot acknowledgment timeout  
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 23 **ACKFAIL**: Boot acknowledgment received (boot acknowledgment check fail)  
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 22 **SDIOIT**: SDIO interrupt received  
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 21 **BUSYD0END**: end of SDMMC_D0 Busy following a CMD response detected  
This indicates only end of busy following a CMD response. This bit does not signal busy due to data transfer. Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.  
0: card SDMMC_D0 signal does NOT signal change from busy to not busy.  
1: card SDMMC_D0 signal changed from busy to NOT busy.

Bit 20 **BUSYD0**: Inverted value of SDMMC_D0 line (Busy), sampled at the end of a CMD response and a second time 2 SDMMC_CK cycles after the CMD response  
This bit is reset to not busy when the SDMMC_D0 line changes from busy to not busy. This bit does not signal busy due to data transfer. This is a hardware status flag only, it does not generate an interrupt.  
0: card signals not busy on SDMMC_D0.  
1: card signals busy on SDMMC_D0.

Bit 19 **RXFIFOE**: Receive FIFO empty  
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes full.

Bit 18 **TXFIFOE**: Transmit FIFO empty  
This bit is cleared when one FIFO location becomes full.

Bit 17 **RXFIFOF**: Receive FIFO full  
This bit is cleared when one FIFO location becomes empty.

Bit 16 **TXFIFOF**: Transmit FIFO full  
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes empty.

Bit 15 **RXFIFOHF**: Receive FIFO half full  
There are at least half the number of words in the FIFO. This bit is cleared when the FIFO becomes half+1 empty.

Bit 14 **TXFIFOHE**: Transmit FIFO half empty  
At least half the number of words can be written into the FIFO. This bit is cleared when the FIFO becomes half+1 full.

Bit 13 **CPSMACT**: Command path state machine active, i.e. not in Idle state  
This is a hardware status flag only, does not generate an interrupt.

Bit 12 **DPSMACT**: Data path state machine active, i.e. not in Idle state  
This is a hardware status flag only, does not generate an interrupt.

Bit 11 **DABORT**: Data transfer aborted by CMD12  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
Bit 10 **DBCKEND**: Data block sent/received
DBCKEND is set when:
- CRC check passed and DPSM moves to the R_W state
- IDMAEN = 0 and transmit data transfer hold and DATACOUNT >0 and DPSM moves to Wait_S.
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 9 **DHO</br>LD**: Data transfer Hold
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 8 **DATAEND**: Data transfer ended correctly
DATAEND is set if data counter DATACOUNT is zero and no errors occur, and no transmit data transfer hold.
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 7 **CMDSENT**: Command sent (no response required)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 6 **CMDREND**: Command response received (CRC check passed, or no CRC)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 5 **RXOVERR**: Received FIFO overrun error (masked by hardware when IDMA is enabled)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 4 **TXUNDERR**: Transmit FIFO underrun error (masked by hardware when IDMA is enabled)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 3 **DTIMEOUT**: Data timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 2 **CTIMEOUT**: Command response timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
The Command Timeout period has a fixed value of 64 SDMMC_CK clock periods.

Bit 1 **DCRCFAIL**: Data block sent/received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 0 **CCRCFAIL**: Command response received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

*Note:* FIFO interrupt flags must be masked in SDMMC_MASKR when using IDMA mode.
24.10.12 SDMMC interrupt clear register (SDMMC_ICR)

Address offset: 0x038

Reset value: 0x0000 0000

The SDMMC_ICR register is a write-only register. Writing a bit with 1 clears the corresponding bit in the SDMMC_STAR status register.

| Bit 31:29 Reserved, must be kept at reset value. |
| Bit 28 IDMABTCC: IDMA buffer transfer complete clear bit |
| Set by software to clear the IDMABTC flag. |
| 0: IDMABTC not cleared |
| 1: IDMABTC cleared |

| Bit 27 IDMATEC: IDMA transfer error clear bit |
| Set by software to clear the IDMATE flag. |
| 0: IDMATE not cleared |
| 1: IDMATE cleared |

| Bit 26 CKSTOPOC: CKSTOP flag clear bit |
| Set by software to clear the CKSTOP flag. |
| 0: CKSTOP not cleared |
| 1: CKSTOP cleared |

| Bit 25 VSWEndC: VSWEND flag clear bit |
| Set by software to clear the VSWEND flag. |
| 0: VSWEND not cleared |
| 1: VSWEND cleared |

| Bit 24 ACKTIMEOUTC: ACKTIMEOUT flag clear bit |
| Set by software to clear the ACKTIMEOUT flag. |
| 0: ACKTIMEOUT not cleared |
| 1: ACKTIMEOUT cleared |

| Bit 23 ACKFAILC: ACKFAIL flag clear bit |
| Set by software to clear the ACKFAIL flag. |
| 0: ACKFAIL not cleared |
| 1: ACKFAIL cleared |

| Bit 22 SDIOITC: SDIOIT flag clear bit |
| Set by software to clear the SDIOIT flag. |
| 0: SDIOIT not cleared |
| 1: SDIOIT cleared |
Bit 21 **BUSYD0ENDC**: BUSYD0END flag clear bit
   Set by software to clear the BUSYD0END flag.
   0: BUSYD0END not cleared
   1: BUSYD0END cleared

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **DABORTC**: DABORT flag clear bit
   Set by software to clear the DABORT flag.
   0: DABORT not cleared
   1: DABORT cleared

Bit 10 **DBCKENDC**: DBCKEND flag clear bit
   Set by software to clear the DBCKEND flag.
   0: DBCKEND not cleared
   1: DBCKEND cleared

Bit 9 **DHOLDC**: DHOLD flag clear bit
   Set by software to clear the DHOLD flag.
   0: DHOLD not cleared
   1: DHOLD cleared

Bit 8 **DATAENDC**: DATAEND flag clear bit
   Set by software to clear the DATAEND flag.
   0: DATAEND not cleared
   1: DATAEND cleared

Bit 7 **CMDSENTC**: CMDSENT flag clear bit
   Set by software to clear the CMDSENT flag.
   0: CMDSENT not cleared
   1: CMDSENT cleared

Bit 6 **CMDRENDC**: CMDREND flag clear bit
   Set by software to clear the CMDREND flag.
   0: CMDREND not cleared
   1: CMDREND cleared

Bit 5 **RXOVERRC**: RXOVERR flag clear bit
   Set by software to clear the RXOVERR flag.
   0: RXOVERR not cleared
   1: RXOVERR cleared

Bit 4 **TXUNDERRC**: TXUNDErr flag clear bit
   Set by software to clear TXUNDErr flag.
   0: TXUNDErr not cleared
   1: TXUNDErr cleared

Bit 3 **DTIMEOUTC**: DTIMEOUT flag clear bit
   Set by software to clear the DTIMEOUT flag.
   0: DTIMEOUT not cleared
   1: DTIMEOUT cleared
24.10.13 SDMMC mask register (SDMMC_MASKR)

Address offset: 0x03C
Reset value: 0x0000 0000

The interrupt mask register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
<th>Description</th>
<th>Reset Value</th>
<th>Bit Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-29</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
<td>0000</td>
<td>rw rw rw rw rw rw rw rw rw rw rc rw</td>
</tr>
<tr>
<td>28</td>
<td>IDMA_TBCTIE</td>
<td>IDMA buffer transfer complete interrupt enable</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>CKSTOPIE</td>
<td>Voltage Switch clock stopped interrupt enable</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>VSWENDIE</td>
<td>Voltage switch critical timing section completion interrupt enable</td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 IDMA_TBCTIE: IDMA buffer transfer complete interrupt enable
Set and cleared by software to enable/disable the interrupt generated when the IDMA has transferred all data belonging to a memory buffer.
0: IDMA buffer transfer complete interrupt disabled
1: IDMA buffer transfer complete interrupt enabled

Bit 27 Reserved, must be kept at reset value.

Bit 26 CKSTOPIE: Voltage Switch clock stopped interrupt enable
Set and cleared by software to enable/disable interrupt caused by Voltage Switch clock stopped.
0: Voltage Switch clock stopped interrupt disabled
1: Voltage Switch clock stopped interrupt enabled

Bit 25 VSWENDIE: Voltage switch critical timing section completion interrupt enable
Set and cleared by software to enable/disable the interrupt generated when voltage switch critical timing section completion.
0: Voltage switch critical timing section completion interrupt disabled
1: Voltage switch critical timing section completion interrupt enabled
Bit 24 **ACKTIMEOUTIE**: Acknowledgment timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment timeout.
0: Acknowledgment timeout interrupt disabled
1: Acknowledgment timeout interrupt enabled

Bit 23 **ACKFAILIE**: Acknowledgment Fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment Fail.
0: Acknowledgment Fail interrupt disabled
1: Acknowledgment Fail interrupt enabled

Bit 22 **SDIOITIE**: SDIO mode interrupt received interrupt enable
Set and cleared by software to enable/disable the interrupt generated when receiving the SDIO mode interrupt.
0: SDIO Mode interrupt received interrupt disabled
1: SDIO Mode interrupt received interrupt enabled

Bit 21 **BUSYD0ENDIE**: BUSYD0END interrupt enable
Set and cleared by software to enable/disable the interrupt generated when SDMMC_D0 signal changes from busy to NOT busy following a CMD response.
0: BUSYD0END interrupt disabled
1: BUSYD0END interrupt enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TXFIFOEIE**: Tx FIFO empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO empty.
0: Tx FIFO empty interrupt disabled
1: Tx FIFO empty interrupt enabled

Bit 17 **RXFIFOFIE**: Rx FIFO full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO full.
0: Rx FIFO full interrupt disabled
1: Rx FIFO full interrupt enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **RXFIFOHFIIE**: Rx FIFO half full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO half full.
0: Rx FIFO half full interrupt disabled
1: Rx FIFO half full interrupt enabled

Bit 14 **TXFIFOHEIE**: Tx FIFO half empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO half empty.
0: Tx FIFO half empty interrupt disabled
1: Tx FIFO half empty interrupt enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **DABORTIE**: Data transfer aborted interrupt enable
Set and cleared by software to enable/disable interrupt caused by a data transfer being aborted.
0: Data transfer abort interrupt disabled
1: Data transfer abort interrupt enabled

Bit 10 **DBCKENDIE**: Data block end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data block end.
0: Data block end interrupt disabled
1: Data block end interrupt enabled
Bit 9 **D HOLDIE**: Data hold interrupt enable
Set and cleared by software to enable/disable the interrupt generated when sending new
data is hold in the DPSM Wait_S state.
0: Data hold interrupt disabled
1: Data hold interrupt enabled

Bit 8 **DATAENDIE**: Data end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data end.
0: Data end interrupt disabled
1: Data end interrupt enabled

Bit 7 **CMDSENTIE**: Command sent interrupt enable
Set and cleared by software to enable/disable interrupt caused by sending command.
0: Command sent interrupt disabled
1: Command sent interrupt enabled

Bit 6 **CMDRENDIE**: Command response received interrupt enable
Set and cleared by software to enable/disable interrupt caused by receiving command
response.
0: Command response received interrupt disabled
1: Command Response received interrupt enabled

Bit 5 **RXOVERRIE**: Rx FIFO overrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO overrun error.
0: Rx FIFO overrun error interrupt disabled
1: Rx FIFO overrun error interrupt enabled

Bit 4 **TXUNDERRIE**: Tx FIFO underrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO underrun error.
0: Tx FIFO underrun error interrupt disabled
1: Tx FIFO underrun error interrupt enabled

Bit 3 **DTIMEOUTIE**: Data timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by data timeout.
0: Data timeout interrupt disabled
1: Data timeout interrupt enabled

Bit 2 **CTIMEOUTIE**: Command timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by command timeout.
0: Command timeout interrupt disabled
1: Command timeout interrupt enabled

Bit 1 **DCRCFAILIE**: Data CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by data CRC failure.
0: Data CRC fail interrupt disabled
1: Data CRC fail interrupt enabled

Bit 0 **CCRCFAILIE**: Command CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by command CRC failure.
0: Command CRC fail interrupt disabled
1: Command CRC fail interrupt enabled
24.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER)

Address offset: 0x040  
Reset value: 0x0000 0000  
The SDMMC_ACKTIMER register contains the acknowledgment timeout period, in SDMMC_CK bus clock periods.

A counter loads the value from the SDMMC_ACKTIMER register, and starts decrementing when the data path state machine (DPSM) enters the Wait_Ack state. If the timer reaches 0 while the DPSM is in this state, the acknowledgment timeout status flag is set.

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

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **ACKTIME[24:0]**: Boot acknowledgment timeout period  
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).  
Boot acknowledgment timeout period expressed in card bus clock periods.

*Note:* The data transfer must be written to the acknowledgment timer register before being written to the data control register.

24.10.15 SDMMC data FIFO registers x (SDMMC_FIFORx)

Address offset: 0x080 + 0x004 * x, (x =0 to 15)  
Reset value: 0x0000 0000  
The receive and transmit FIFOs can be only read or written as word (32-bit) wide registers. The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

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

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.
Bits 31:0 **FIFODATA[31:0]**: Receive and transmit FIFO data
This register can only be read or written by firmware when the DPSM is active (DPSMACT = 1).
The FIFO data occupies 16 entries of 32-bit words.

### 24.10.16 SDMMC DMA control register (SDMMC_IDMACTRLR)

Address offset: 0x050
Reset value: 0x0000 0000

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 32 entries on 32 sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO.

<table>
<thead>
<tr>
<th>Bits 31:2</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1</td>
<td><strong>IDMABMODE</strong>: Buffer mode selection</td>
</tr>
<tr>
<td></td>
<td>This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).</td>
</tr>
<tr>
<td></td>
<td>0: Single buffer mode.</td>
</tr>
<tr>
<td></td>
<td>1: Linked list mode.</td>
</tr>
<tr>
<td>Bit 0</td>
<td><strong>IDMAEN</strong>: IDMA enable</td>
</tr>
<tr>
<td></td>
<td>This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).</td>
</tr>
<tr>
<td></td>
<td>0: IDMA disabled</td>
</tr>
<tr>
<td></td>
<td>1: IDMA enabled</td>
</tr>
</tbody>
</table>

### 24.10.17 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER)

Address offset: 0x054
Reset value: 0x0000 0000

The SDMMC_IDMABSIZER register contains the buffer size when in linked list configuration.

<table>
<thead>
<tr>
<th>Bits 31:2</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1</td>
<td><strong>IDMABNDT</strong>: Buffer mode selection</td>
</tr>
<tr>
<td></td>
<td>This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).</td>
</tr>
<tr>
<td></td>
<td>0: Single buffer mode.</td>
</tr>
<tr>
<td></td>
<td>1: Linked list mode.</td>
</tr>
<tr>
<td>Bit 0</td>
<td><strong>IDMAEN</strong>: IDMA enable</td>
</tr>
<tr>
<td></td>
<td>This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).</td>
</tr>
<tr>
<td></td>
<td>0: IDMA disabled</td>
</tr>
<tr>
<td></td>
<td>1: IDMA enabled</td>
</tr>
</tbody>
</table>
24.10.18  SDMMC IDMA buffer base address register (SDMMC_IDMABASER)

Address offset: 0x058

Reset value: 0x0000 0000

The SDMMC_IDMABASER register contains the memory buffer base address in single buffer configuration and linked list configuration.

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x058</td>
<td>IDMABASE[31:16]</td>
<td>rw rw</td>
<td>Bits 31:17: Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td></td>
<td>IDMABASE[15:0]</td>
<td>rw rw</td>
<td>Bits 16:5 IDMABNDT[11:0]: Number of bytes per buffer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This 12-bit value must be multiplied by 8 to get the size of the buffer in 32-bit words and by 32 to get the size of the buffer in bytes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: IDMABNDT = 0x001: buffer size = 8 words = 32 bytes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: IDMABNDT = 0x800: buffer size = 16384 words = 64 Kbyte.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).</td>
</tr>
<tr>
<td></td>
<td>IDMABASE[15:0]</td>
<td>rw rw</td>
<td>Bits 4:0: Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

24.10.19  SDMMC IDMA linked list address register (SDMMC_IDMALAR)

Address offset: 0x064

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x064</td>
<td>IDMALA[13:0]</td>
<td>rw rw</td>
<td>Bits 31:0: IDMABASE[31:0]: Buffer memory base address bits [31:2], must be word aligned (bits [1:0] are always 0 and read only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1).</td>
</tr>
</tbody>
</table>

RM0481  Secure digital input/output MultiMediaCard interface (SDMMC)
Bit 31 **ULA**: Update SDMMC_IDMALAR from linked list when in linked list mode
(SDMMC_IDMACTRLR.IDMABMODE select linked list mode)

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: SDMMC_IDMALAR is not to be updated, last linked list item.
1: SDMMC_IDMALAR is to be updated from linked list table.

Bit 30 **ULS**: Update SDMMC_IDMABSIZE from the next linked list when in linked list mode
(SDMMC_IDMACTRLR.IDMABMODE select linked list mode and ULA = 1)

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: SDMMC_IDMABSIZE is not to be updated from next linked list table.
1: SDMMC_IDMABSIZE is to be updated from next linked list table.

Bit 29 **ABR**: Acknowledge linked list buffer ready

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
This bit is not taken into account when starting the first linked list buffer from the software
programmed register information. ABR is only taken into account on subsequent loaded
linked list items.
0: Loaded linked list buffer is not ready (this causes a linked list IDMA transfer error to be
     generated).
1: Loaded linked list buffer ready acknowledge. Linked list buffer data are transfered by
    IDMA.

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:2 **IDMALA[13:0]**: Word aligned linked list item address offset

Linked list item offset pointer to the base of the next linked list item structure.
Linked list item base address is IDMABA + IDMALA.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

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

### 24.10.20 SDMMC IDMA linked list memory base register
(SDMMC_IDMABAR)

Address offset: 0x068
Reset value: 0x0000 0000

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

Bits 31:2 **IDMABA[29:0]**: Word aligned Linked list memory base address

Linked list memory base pointer.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 1:0 Reserved, must be kept at reset value.
## 24.10.21 SDMMC register map

### Table 235. SDMMC register map

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | SDMMC_ POWER  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x004  | SDMMC_ CLKCR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x008  | SDMMC_ ARGR_ |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | CMDARG[31:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x00C  | SDMMC_ CMDR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x010  | SDMMC_ RESPCMDR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x014  | SDMMC_ RESP1R |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | CARDSTATUS[31:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x018  | SDMMC_ RESP2R |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | CARDSTATUS[31:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x01C  | SDMMC_ RESP3R |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | CARDSTATUS[31:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x020  | SDMMC_ RESP4R |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | CARDSTATUS[31:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x024  | SDMMC_ DTIMER |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | DATETIME[31:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x028  | SDMMC_ DLENR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | DATALENGTH[24:0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x02C  | SDMMC_ DCTRLR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Res.          | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
Table 235. SDMMC register map (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x030</td>
<td>SDMMC_DCNTRR</td>
<td>DATACOUNT[24:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x034</td>
<td>SDMMC_STAR</td>
<td>IDMAMCDesigner</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x038</td>
<td>SDMMC_ICR</td>
<td>IDMABTCC</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x03C</td>
<td>SDMMC_MASKR</td>
<td>IDMABTCIE</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x040</td>
<td>SDMMC_ACKTIMER</td>
<td>ACKTIME[24:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x044</td>
<td></td>
<td>Reserved</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x045</td>
<td></td>
<td>SDMMC_IDMAMCR</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x050</td>
<td></td>
<td>IDMABNDR</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x054</td>
<td></td>
<td>IDMABCD[11:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x058</td>
<td></td>
<td>IDMABASE[31:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x05C</td>
<td></td>
<td>Reserved</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x060</td>
<td></td>
<td>SDMMC_IDMALAR</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x064</td>
<td></td>
<td>IDMALL[13:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x068</td>
<td></td>
<td>IDMABA[28:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x06C</td>
<td></td>
<td>Reserved</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x070</td>
<td></td>
<td>SDMMC_FIFOR</td>
<td>FIFODATA[31:0]</td>
</tr>
</tbody>
</table>

Offset values are in hexadecimal and range from 0x0000 to 0x07C.
Refer to Section 2.3 on page 111 for the register boundary addresses.
25 Delay block (DLYB)

25.1 Introduction

The delay block (DLYB) is used to generate an output clock that is dephased from the input clock. The phase of the output clock must be programmed by the user application. The output clock is then used to clock the data received by another peripheral such as an SDMMC or Octo-SPI interface.

The delay is voltage- and temperature-dependent, that may require the application to reconfigure and recenter the output clock phase with the receive data.

25.2 DLYB main features

The delay block has the following features:

- Input clock frequency ranging from 25 MHz to the maximum frequency supported by the communication interface (see datasheet)
- Up to 12 oversampling phases.

25.3 DLYB functional description

25.3.1 DLYB diagram

The delay block includes the following sub-blocks (shown in the figure below):

- register interface block providing AHB access to the DLYB registers
- delay line supporting the unit delays
- delay line length sampling
- output clock selection multiplexer

![Figure 186. DLYB block diagram]
25.3.2 DLYB pins and internal signals

Table 236 lists the DLYB internal signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dlyb_hclk</td>
<td>Digital input</td>
<td>Delay block register interface clock</td>
</tr>
<tr>
<td>dlyb_in_ck</td>
<td>Digital input</td>
<td>Delay block input clock</td>
</tr>
<tr>
<td>dlyb_out_ck</td>
<td>Digital output</td>
<td>Delay block output clock</td>
</tr>
</tbody>
</table>

25.3.3 General description

The delay block is enabled by setting the DEN bit in the DLYB control register (DLYB_CR). The length sampler is enabled through the SEN bit in DLYB_CR register.

When the delay block is enabled, the delay added by a unit delay is defined by the UNIT[6:0] field in the DLYB configuration register (DLYB_CFGR).

**Note:** UNIT[6:0] can be programmed only when the output clock is disabled (SEN = 1).

When the delay block is enabled, the output clock phase is selected through the SEL[3:0] field in DLYB_CFGR register.

**Note:** SEL can be programmed only when the output clock is disabled (SEN = 1).

The output clock can be de-phased over one input clock period by configuring the delay line length to span one period. The delay line length can be configured by enabling the length sampler through the SEN bit, that gives access to the delay line length (LNG[11:0]) and length valid flag (LNGF) in DLYB_CFGREG.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced. This allows a smaller unit delay providing higher resolution.

Once the delay line length is configured, a dephased output clock can be selected by the output clock multiplexer. This is done through SEL[3:0]. The output clock is only available on the selected phase when SEN is set to 0.

The table below gives a summary of the delay block control.

<table>
<thead>
<tr>
<th>DEN</th>
<th>SEN</th>
<th>UNIT</th>
<th>SEL</th>
<th>LNG</th>
<th>LNGF</th>
<th>Output clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Enabled (≠ Input clock)</td>
</tr>
<tr>
<td>x</td>
<td>1</td>
<td>Unit delay</td>
<td>Output clock phase</td>
<td>Length</td>
<td>Length flag</td>
<td>Disabled</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Unit delay(1)</td>
<td>Output clock phase(2)</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Enabled (≠ selected phase)</td>
</tr>
</tbody>
</table>

1. The unit delay can only be changed when SEN = 1.
2. The output clock phase can only be changed when SEN = 1.
25.3.4 Delay line length configuration procedure

LNG[11:0] is used to determine the delay line length with respect to the input clock period. The length must be configured so that one full input clock period is covered by the delay line length.

Note that despite the delay line has 12 unit delay elements, the following procedure description returns a length between 0 and 10, as the upper delay output value is used to ensure that the delay is calibrated over one full input clock cycle. Depending on the clock frequency and UNIT value, unit delay element 10 may also be truncated from the clock cycle length.

A clock input (free running clock) must be present during the whole tuning procedure.

To configure the delay line length to one period of the Input clock, follow the sequence below:

1. Enable the delay block by setting DEN bit to 1.
2. Enable the length sampling by setting SEN bit to 1.
3. Enable all delay cells by setting SEL[3:0] to 12.
4. For UNIT[6:0] = 0 to 127 (this step must be repeated until the delay line length is configured):
   a) Update the UNIT[6:0] value and wait till the length flag LNGF is set to 1.
   b) Read LNG[11:0].
      If (LNG[10:0] > 0) and (LNG[11] or LNG[10] = 0), the delay line length is configured to one input clock period.
5. Determine how many unit delays (N) span one input clock period: for N = 0 to 10, if LNG[N] = 1, the number of unit delays spanning the input clock period = N.
6. Disable the length sampling by clearing SEN to 0.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced smaller than one input clock period. This allows a smaller unit delay, providing a higher resolution spanning a shorter time interval.

25.3.5 Output clock phase configuration procedure

When the delay line length is configured to one input clock period, the output clock phase can be selected between the unit delays spanning one Input clock period.

Follow the steps below to select the output clock phase:

1. Disable the output clock and enable the access to the phase selection SEL[3:0] bits by setting SEN bit to 1.
2. Program SEL[3:0] with the desired output clock phase value.
3. Enable the output clock on the selected phase by clearing SEN to 0.

Octo-SPI use case:

The delay block is used in conjunction with Octo-SPI interface to allow shifting the input data sampling signal. This sampling signal can be the feedback clock or the data strobe (DQS) signal, which is delivered by certain type of devices. Note that in case DQS is used, the calibration procedure must be performed beforehand with a free running clock, as DQS is a discontinuous signal.

In case of SDR (single data rate) mode the user must typically shift the sampling signal by half period, so that the sampling edges are positioned in the middle of the valid data phase.
In case of DDR (dual data rate) mode, for which data are transitioning at start and middle of period, typical value must be close to N/4, once the calibration is completed.

In case of high frequencies and tight timing constraints, the delay setting granularity (10) might be too coarse. Since in most cases it is not necessary to have a possible delay value covering the whole sampling clock period, the "Unit" value can be overridden by application in order to improve the accuracy of the sampling edge position (example: providing a twice as small "Unit" gives twice better timing accuracy. The counterpart being that the maximum possible delay is divided by 2).

SDMMC use case:

The delay block is used in conjunction with SDMMC interface variable delay. For correct sampling point tuning the delay value must cover a whole SDMMC_CK clock period. After having tuned the delay line length the individual delays are used in the sampling point tuning to find the optimal sampling point.

25.4 DLYB registers

All registers can be accessed in word, half-word and byte access.

25.4.1 DLYB control register (DLYB_CR)

Address offset: 0x000
Reset value: 0x0000 0000

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

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

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

Bit 1 **SEN**: Sampler length enable bit
0: Sampler length and register access to UNIT[6:0] and SEL[3:0] disabled, output clock enabled.
1: Sampler length and register access to UNIT[6:0] and SEL[3:0] enabled, output clock disabled.

Bit 0 **DEN**: Delay block enable bit
0: DLYB disabled.
1: DLYB enabled.
25.4.2 DLYB configuration register (DLYB_CFRG)

Address offset: 0x004
Reset value: 0x00000000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
</table>
| 31  | LNGF: Length valid flag | This flag indicates when the delay line length value contained in LNG[11:0] is valid after UNIT[6:0] bits changed.  
0: Length value in LNG is not valid.  
1: Length value in LNG is valid. |
| 30-28 | Reserved | Must be kept at reset value. |
| 27-16 | LNG[11:0]: Delay line length value | These bits reflect the 12 unit delay values sampled at the rising edge of the input clock.  
The value is only valid when LNGF = 1. |
| 15  | Reserved | Must be kept at reset value. |
| 14-8 | UNIT[6:0]: Delay of a unit delay cell. | These bits can only be written when SEN = 1.  
Unit delay = initial delay + UNIT[6:0] x delay step |
| 7-4  | Reserved | Must be kept at reset value. |
| 3-0  | SEL[3:0]: Phase for the output clock. | These bits can only be written when SEN = 1.  
Output clock phase = input clock + SEL[3:0] x unit delay |

25.4.3 DLYB register map

Table 238. DLYB register map and reset values

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>DLYB_CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Register name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x004</td>
<td>DLYB_CFRG</td>
<td>LNGF</td>
<td>LNG[11:0]</td>
<td>UNIT[6:0]</td>
<td>SEL[3:0]</td>
</tr>
<tr>
<td></td>
<td>0x00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 111 for the register boundary addresses.
26  Analog-to-digital converters (ADC1/2)

26.1  Introduction

This section describes the implementation of up to 2 ADCs:
- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master).

Each ADC consists of a 12-bit successive approximation analog-to-digital converter.

Each ADC has up to 20 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows improving analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

26.2  ADC main features

- High-performance features
  - Up to 2 ADCs which can operate in dual mode:
    - ADC1 is connected to 18 external channels and to 2 internal channels
    - ADC2 is connected to 18 external channels and to 2 internal channels
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Manage single-ended or differential inputs
  - AHB slave bus interface to allow fast data handling
  - Self-calibration
  - Channel-wise programmable sampling time
  - Flexible sampling time control
  - Up to 4 injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - Four dedicated data registers for the injected channels

- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
Analog-to-digital converters (ADC1/2)

- Allows slow bus frequency application while keeping optimum ADC performance
- Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (auto-delayed mode)

- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256x
  - Programmable data shift up to 8 bits

- Data preconditioning
  - Offset compensation

- Analog input channels
  - External analog inputs (per ADC):
    - Up to 6 fast channels from GPIO pads
    - Up to 12 slow channels from GPIO pads
  - 1 channel for the internal temperature sensor ($V_{SENSE}$)
  - 1 channel for the internal reference voltage ($V_{REFINT}$)
  - 1 channel for monitoring the external VBAT power supply pin
  - 1 channel for monitoring the internal $V_{DDCORE}$ supply

- Start-of-conversion can be initiated:
  - By software for both regular and injected conversions
  - By hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions

- Conversion modes
  - Each ADC can convert a single channel or can scan a sequence of channels
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode

- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events

- 3 analog watchdogs per ADC
  - Watchdog can perform filtering to ignore out-of-range data

- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{REF+}$

*Figure 187* shows the block diagram of one ADC.
26.3 ADC implementation

<table>
<thead>
<tr>
<th>ADC modes/features</th>
<th>ADC1</th>
<th>ADC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>12 bit</td>
<td>5 Msps (12-bit resolution)</td>
</tr>
<tr>
<td>Maximum sampling speed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dual mode operation</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Hardware offset calibration</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Hardware linearity calibration</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Single-end input</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Differential input</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Injected channel conversion</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Oversampling</td>
<td>up to x256</td>
<td></td>
</tr>
<tr>
<td>Data register</td>
<td>16 bits</td>
<td></td>
</tr>
<tr>
<td>Data register FIFO depth</td>
<td>3 stages</td>
<td></td>
</tr>
<tr>
<td>DMA support</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Parallel data output to ADF</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Offset compensation</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Gain compensation</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Number of analog watchdog</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Option register</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
26.4  ADC functional description

26.4.1  ADC block diagram

Figure 187 shows the ADC block diagram and Table 240 gives the ADC pin description.
### 26.4.2 ADC pins and internal signals

#### Table 240. ADC input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDDA</td>
<td>Input, analog supply</td>
<td>Analog power supply and positive reference voltage for the ADC</td>
</tr>
<tr>
<td>VSSA</td>
<td>Input, analog supply ground</td>
<td>Ground for analog power supply, equal to VSS.</td>
</tr>
<tr>
<td>VREF+</td>
<td>Input, analog reference positive</td>
<td>The higher/positive reference voltage for the ADC.</td>
</tr>
<tr>
<td>VREF−</td>
<td>Input, analog reference negative</td>
<td>The lower/negative reference voltage for the ADC.</td>
</tr>
<tr>
<td>ADC1/2_INNi/INPi</td>
<td>Negative/positive external analog input signals</td>
<td>20 negative/positive external analog input channels (refer to Section 26.4.4: ADC connectivity for details)</td>
</tr>
</tbody>
</table>

#### Table 241. ADC internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VINPi</td>
<td>Positive analog input channels</td>
<td>Positive internal analog input channels connected either to ADC1/2_INPi/ external channels or to internal channels.</td>
</tr>
<tr>
<td>VINNi</td>
<td>Negative analog input channels</td>
<td>Negative internal analog input channels connected either to ADC1/2_INNi/ external channels or to internal channels.</td>
</tr>
<tr>
<td>adc_ext_trgi</td>
<td>Inputs</td>
<td>ADC external trigger inputs for regular conversions. These inputs are shared between the ADC master and the ADC slave.</td>
</tr>
<tr>
<td>adc_jext_trgi</td>
<td>Inputs</td>
<td>ADC external trigger inputs for the injected conversions. These inputs are shared between the ADC master and the ADC slave.</td>
</tr>
<tr>
<td>adc_awdx</td>
<td>Output</td>
<td>Internal analog watchdog output signal connected to on-chip timers. (x = Analog watchdog number 1,2,3)</td>
</tr>
<tr>
<td>adc_ker_ck_input</td>
<td>Output</td>
<td>ADC kernel clock</td>
</tr>
<tr>
<td>adc_hclk</td>
<td>Input</td>
<td>ADC peripheral clock</td>
</tr>
<tr>
<td>adc_it</td>
<td>Output</td>
<td>ADC interrupt</td>
</tr>
<tr>
<td>adc_dma</td>
<td>Output</td>
<td>ADC DMA request</td>
</tr>
</tbody>
</table>

#### Table 242. ADC interconnection

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source/destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC1 VINp[16]</td>
<td>VSENSE (internal temperature sensor output voltage).</td>
</tr>
<tr>
<td>ADC1 VINp[17]</td>
<td>VREFINT (output voltage from internal reference voltage).</td>
</tr>
<tr>
<td>ADC2 VINp[16]</td>
<td>VBAT/4 (VBAT pin input voltage divided by 4).</td>
</tr>
<tr>
<td>ADC2 VINp[17]</td>
<td>VDDCORE (internal digital core voltage).</td>
</tr>
<tr>
<td>adc_ext_trg0</td>
<td>tim1_oc1</td>
</tr>
<tr>
<td>Signal name</td>
<td>Source/destination</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>adc_ext_trg1</td>
<td>tim1_oc2</td>
</tr>
<tr>
<td>adc_ext_trg2</td>
<td>tim1_oc3</td>
</tr>
<tr>
<td>adc_ext_trg3</td>
<td>tim2_oc2</td>
</tr>
<tr>
<td>adc_ext_trg4</td>
<td>tim3_trgo</td>
</tr>
<tr>
<td>adc_ext_trg5</td>
<td>tim4_oc4</td>
</tr>
<tr>
<td>adc_ext_trg6</td>
<td>exti11</td>
</tr>
<tr>
<td>adc_ext_trg7</td>
<td>tim8_trgo</td>
</tr>
<tr>
<td>adc_ext_trg8</td>
<td>tim8_trgo2</td>
</tr>
<tr>
<td>adc_ext_trg9</td>
<td>tim1_trgo</td>
</tr>
<tr>
<td>adc_ext_trg10</td>
<td>tim1_trgo2</td>
</tr>
<tr>
<td>adc_ext_trg11</td>
<td>tim2_trgo</td>
</tr>
<tr>
<td>adc_ext_trg12</td>
<td>tim4_trgo</td>
</tr>
<tr>
<td>adc_ext_trg13</td>
<td>tim6_trgo</td>
</tr>
<tr>
<td>adc_ext_trg14</td>
<td>tim15_trgo</td>
</tr>
<tr>
<td>adc_ext_trg15</td>
<td>tim3_oc4</td>
</tr>
<tr>
<td>adc_ext_trg16</td>
<td>exti15</td>
</tr>
<tr>
<td>adc_ext_trg17</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg18</td>
<td>lptim1_ch1</td>
</tr>
<tr>
<td>adc_ext_trg19</td>
<td>lptim2_ch1</td>
</tr>
<tr>
<td>adc_ext_trg20</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg21</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg22</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg23</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg24</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg25</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg26</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg27</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg28</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg29</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg30</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg31</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg0</td>
<td>tim1_trgo</td>
</tr>
<tr>
<td>adc_jext_trg1</td>
<td>tim1_oc4</td>
</tr>
<tr>
<td>adc_jext_trg2</td>
<td>tim2_trgo</td>
</tr>
<tr>
<td>adc_jext_trg3</td>
<td>tim2_oc1</td>
</tr>
<tr>
<td>Signal name</td>
<td>Source/destination</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>adc_jext_trg4</td>
<td>tim3_oc4</td>
</tr>
<tr>
<td>adc_jext_trg5</td>
<td>tim4_trgo</td>
</tr>
<tr>
<td>adc_jext_trg6</td>
<td>exti15</td>
</tr>
<tr>
<td>adc_jext_trg7</td>
<td>tim8_oc4</td>
</tr>
<tr>
<td>adc_jext_trg8</td>
<td>tim1_trgo2</td>
</tr>
<tr>
<td>adc_jext_trg9</td>
<td>tim8_trgo</td>
</tr>
<tr>
<td>adc_jext_trg10</td>
<td>tim8_trgo2</td>
</tr>
<tr>
<td>adc_jext_trg11</td>
<td>tim3_oc3</td>
</tr>
<tr>
<td>adc_jext_trg12</td>
<td>tim3_trgo</td>
</tr>
<tr>
<td>adc_jext_trg13</td>
<td>tim3_oc1</td>
</tr>
<tr>
<td>adc_jext_trg14</td>
<td>tim6_trgo</td>
</tr>
<tr>
<td>adc_jext_trg15</td>
<td>tim15_trgo</td>
</tr>
<tr>
<td>adc_jext_trg16</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg17</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg18</td>
<td>lptim1_ch1</td>
</tr>
<tr>
<td>adc_jext_trg19</td>
<td>lptim2_ch1</td>
</tr>
<tr>
<td>adc_jext_trg20</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg21</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg22</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg23</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg24</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg25</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg26</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg27</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg28</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg29</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg30</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg31</td>
<td>reserved</td>
</tr>
</tbody>
</table>
26.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC clock is independent from the AHB bus clock.

The ADC input clock can be selected between two different clock sources (see Figure 188: ADC clock scheme):

1. The ADC clock can be a specific clock source (adc_ker_ck_input), independent and asynchronous with the AHB clock.
   Refer to section Reset and clock control (RCC) for more information on how to generate the ADC dedicated clock. To select this scheme, CKMODE[1:0] bits of ADC_CCR register must be set to 00.
2. The ADC clock can be derived from the AHB clock interface divided by a programmable factor of 1, 2 or 4. To select this scheme, CKMODE[1:0] bits of ADC_CCR must be different from 00. The programmable divider factor can be configured through to CKMODE[1:0] bits of ADC_CCR.
   The prescaling factor of 1 (CKMODE[1:0] = 01) can be used only if the AHB prescaler is set to 1 (HPRE[3:0] = 0xx in the RCC_CFG register).

Option 1 has the advantage of achieving the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can eventually be divided by the following ratio: 1, 2, 4, 6, 8, 12, 16, 32, 64, 128, 256, using the prescaler configured with bits PRESC[3:0] in the ADC_CCR register.

Option 2 has the advantage of bypassing the clock domain resynchronizations. This can be useful when the ADC is triggered by a timer and if the application requires that the ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant is added by the resynchronizations between the two clock domains).

The clock is configured through CKMODE[1:0] bits must be compliant with the operating frequency specified in the datasheet.
Clock ratio constraint between ADC clock and AHB clock

There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- \( F_{adc_hclk} \geq F_{ADC} / 4 \) if the resolution of all channels are 12-bit or 10-bit
- \( F_{adc_hclk} \geq F_{ADC} / 3 \) if there are some channels with resolutions equal to 8-bit (and none with lower resolutions)
- \( F_{adc_hclk} \geq F_{ADC} / 2 \) if there are some channels with resolutions equal to 6-bit

Constraints between ADC clocks

When several ADC interfaces are used simultaneously, it is mandatory to use the same clock source from the RCC block without prescaler ratio for all ADC interfaces.
26.4.4 ADC connectivity

ADC inputs are connected to the external channels as well as internal sources as described below.

Figure 189. ADC1 connectivity
Figure 190. ADC2 connectivity

ADC2

Channel selection

VREF+

ADC12_INP0
ADC12_INN1

ADC12_INP1

ADC2_INP2

ADC12_INP3

ADC12_INP4

ADC12_INP5

ADC2_INP6
ADC2_INN2

ADC12_INP7
ADC12_INN3

ADC12_INP8
ADC12_INN4

ADC12_INP9
ADC12_INN5

ADC12_INP10

ADC12_INP11

ADC12_INP12
ADC12_INN11

ADC12_INP13
ADC12_INN12

ADC12_INP14

ADC12_INP15

ADC12_INP16

ADC12_INP17

ADC12_INN18

ADC12_INP18

ADC12_INP19

ADC12_INN19

VSSA

VREF-

Vinp[0]

Vinp[1]

Vinp[2]

Vinp[3]

Vinp[4]

Vinp[5]

Vinp[6]

Vinp[7]

Vinp[8]

Vinp[9]

Vinp[10]

Vinp[11]

Vinp[12]

Vinp[13]

Vinp[14]

Vinp[15]

Vinp[16]

Vinp[17]

Vinp[18]

Vinp[19]
26.4.5 **Slave AHB interface**

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:
- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

26.4.6 **ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)**

By default, the ADC is in Deep-power-down mode where its supply is internally switched off to reduce the leakage currents (the reset state of bit DEEPPWD is 1 in the ADC_CR register).

To start ADC operations, it is first needed to exit Deep-power-down mode by setting bit DEEPPWD = 0.

Then, it is mandatory to enable the ADC internal voltage regulator by setting the bit ADVREGEN = 1 into ADC_CR register. The software must wait for the startup time of the ADC voltage regulator ($T_{\text{ADCVREG_STUP}}$) before launching a calibration or enabling the ADC. This delay must be implemented by software.

For the startup time of the ADC voltage regulator, refer to device datasheet for $T_{\text{ADCVREG_STUP}}$ parameter.

When ADC operations are complete, the ADC can be disabled (ADEN = 0). It is possible to save power by also disabling the ADC voltage regulator. This is done by writing bit ADVREGEN = 0.

Then, to save more power by reducing the leakage currents, it is also possible to re-enter in ADC Deep-power-down mode by setting bit DEEPPWD = 1 into ADC_CR register. This is particularly interesting before entering Stop mode.

*Note:* *Writing DEEPPWD = 1 automatically disables the ADC voltage regulator and bit ADVREGEN is automatically cleared.*

*When the internal voltage regulator is disabled (ADVREGEN = 0), the internal analog calibration is kept.*

In ADC Deep-power-down mode (DEEPPWD = 1), the internal analog calibration is lost and it is necessary to either relaunch a calibration or re-apply the calibration factor which was previously saved (refer to *Section 26.4.8: Calibration (ADCAL, ADCALDIFF, ADC_CALFACT)*).
26.4.7 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by programming DIFSEL[i] bits in the ADC_DIFSEL register. This configuration must be written while the ADC is disabled (ADEN = 0). Note that the DIFSEL[i] bits corresponding to single-ended channels are always programmed at 0.

In single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage V_{INP[i]} (positive input) and V_{REF-} (negative input).

In differential input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage V_{INP[i]} (positive input) and V_{INN[i]} (negative input).

The output data for the differential mode is an unsigned data. When V_{INP[i]} equals V_{REF-}, V_{INN[i]} equals V_{REF+} and the output data is 0x000 (12-bit resolution mode). When V_{INP[i]} equals V_{REF+}, V_{INN[i]} equals V_{REF-} and the output data is 0xFFF.

$$\text{Converted value} = \frac{\text{ADC Full \ Scale}}{2} \times \left[ 1 + \frac{V_{\text{INP}} - V_{\text{INN}}}{V_{\text{REF+}}} \right]$$

When ADC is configured as differential mode, both inputs should be biased at (V_{REF+})/2 voltage.

The input signals are supposed to be differential (common mode voltage should be fixed).

Internal channels (such as V_{REFINT} and V_{SENSE}) are used in single-ended mode only.

For a complete description of how the input channels are connected for each ADC, refer to Section 26.4.4: ADC connectivity.

Caution: When configuring the channel “i” in differential input mode, its negative input voltage V_{INN[i]} is connected to another channel. As a consequence, this channel is no longer usable in single-ended mode or in differential mode and must never be configured to be converted. Some channels are shared between ADC1/ADC2: this can make the channel on the other ADC unusable. Only exception is interleaved mode for ADC master and the slave.

26.4.8 Calibration (ADCAL, ADCALDIF, ADC_CALFACT)

Each ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor which is 7-bit wide and which is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or bandgap variation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALDIF = 0 before launching a calibration which will be applied for single-ended input conversions.
- Write ADCALDIF = 1 before launching a calibration which will be applied for differential input conversions.

The calibration is then initiated by software by setting bit ADCAL = 1. Calibration can only be initiated when the ADC is disabled (when ADEN = 0). ADCAL bit stays at 1 during all the
calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[6:0] or CALFACT_D[6:0] of ADC_CALFACT register (depending on single-ended or differential input calibration)

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling the ADC.

The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in Standby or VBAT mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADC_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0 and JADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when $V_{REF+}$ voltage changed more than 10%.

**Software procedure to calibrate the ADC**

1. Ensure DEEPPWD = 0, ADVREGEN = 1 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN = 0.
3. Select the input mode for this calibration by setting ADCALDIF = 0 (single-ended input) or ADCALDIF = 1 (differential input).
4. Set ADCAL = 1.
5. Wait until ADCAL = 0.
6. The calibration factor can be read from ADC_CALFACT register.

![Figure 191. ADC calibration](image-url)
Software procedure to reinject a calibration factor into the ADC

1. Ensure ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new calibration factors.
3. When a conversion is launched, the calibration factor is injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

Figure 192. Updating the ADC calibration factor

Converting single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF = 0 and one with ADCALDIF = 1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF = 0). This updates the register CALFACT_S[6:0].
3. Calibrate the ADC in differential input modes (with ADCALDIF = 1). This updates the register CALFACT_D[6:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration is automatically injected into the analog ADC.
26.4.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure explained in Section 26.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)).

Once DEEPPWD = 0 and ADVREGEN = 1, the ADC can be enabled and the ADC needs a stabilization time of \( t_{\text{STAB}} \) before it starts converting accurately, as shown in Figure 194. Two control bits enable or disable the ADC:

- \( \text{ADEN} = 1 \) enables the ADC. The flag ADRDY is set once the ADC is ready for operation.
- \( \text{ADDIS} = 1 \) disables the ADC. ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversion can then start either by setting \( \text{ADSTART} = 1 \) (refer to Section 26.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting \( \text{JADSTART} = 1 \) or when an external injected trigger event occurs, if injected triggers are enabled.

### Software procedure to enable the ADC

1. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’.
2. Set \( \text{ADEN} = 1 \).
3. Wait until ADRDY = 1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE = 1).
4. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’ (optional).

**Caution:** ADEN bit cannot be set when ADCAL is set and during four ADC clock cycles after the ADCAL bit is cleared by hardware (end of the calibration).
Software procedure to disable the ADC

1. Check that both ADSTART = 0 and JADSTART = 0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP = 1 and JADSTP = 1 and then wait until ADSTP = 0 and JADSTP = 0.
2. Set ADDIS = 1.
3. If required by the application, wait until ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

Figure 194. Enabling / disabling the ADC

26.4.10 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC Section), the DIFSEL[i] control bits in the ADC_DIFSEL register and the control bits ADCAL and ADEN in the ADC_CR register, only if the ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADC_CFGR, ADC_SMPRx, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy, ADC_OFCHRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if the ADC is enabled (ADEN = 1) and if there is no regular conversion ongoing (ADSTART must be equal to 0).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if the ADC is enabled (ADEN = 1) and if there is no injected conversion ongoing (JADSTART must be equal to 0).
- ADC_TRy registers can be modified when an analog-to-digital conversion is ongoing (refer to Section 26.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD1HTx, AWD2LTx, AWDx) for details).

The software is allowed to write the ADSTP or JADSTP control bits of the ADC_CR register only if the ADC is enabled, possibly converting, and if there is no pending request to disable the ADC (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).
The software can write the register ADC_JSQR at any time, when the ADC is enabled (ADEN = 1). Refer to Section 26.6.16: ADC injected sequence register (ADC_JSQR) for additional details.

Note: There is no hardware protection to prevent these forbidden write accesses and ADC behavior may become in an unknown state. To recover from this situation, the ADC must be disabled (clear ADEN = 0 as well as all the bits of ADC_CR register).

26.4.11 Channel selection (SQRx, JSQRx)

The ADC features up to 20 multiplexed channels per ADC, out of which:

- Up to 18 analog inputs coming from GPIO pads (ADC_INP/INN[i]) depending on the products, not all of them are available on GPIO pads.
- ADC is connected to 4 internal analog inputs:
  - the internal temperature sensor (VSENSE)
  - the internal reference voltage (VREFINT)
  - the VBAT monitoring channel (VBAT/4)
  - the internal digital core voltage (VDDCORE)

To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming bits VREFEN, VBATEN or TSEN in the ADC_CCR registers.

Refer to Table ADC interconnection in Section 26.4.2: ADC pins and internal signals for the connection of the above internal analog inputs to external ADC pins or internal signals.

The conversions can be organized in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADC1/2_INP/INN3, ADC1/2_INP/INN8, ADC1/2_INP/INN2, ADC1/2_INN/INP2, ADC1/2_INP/INN0, ADC1/2_INP/INN2, ADC1/2_INP/INN2, ADC1/2_INP/INN15.

- A regular group is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.

- An injected group is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while regular conversions can occur. For this, the ADC regular conversions must be first stopped by writing ADSTP = 1 (refer to Section 26.4.17: Stopping an ongoing conversion (ADSTP, JADSTP)).

The software is allowed to modify on-the-fly the ADC_JSQR register when JADSTART is set to 1 (injected conversions ongoing) only when the context queue is enabled (JQDIS = 0 in ADC_CFRG register). Refer to Section 26.4.21: Queue of context for injected conversions...
26.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADC_SMPR1 and ADC registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 2.5 ADC clock cycles
- SMP = 001: 6.5 ADC clock cycles
- SMP = 010: 12.5 ADC clock cycles
- SMP = 011: 24.5 ADC clock cycles
- SMP = 100: 47.5 ADC clock cycles
- SMP = 101: 92.5 ADC clock cycles
- SMP = 110: 247.5 ADC clock cycles
- SMP = 111: 640.5 ADC clock cycles

The total conversion time is calculated as follows:

\[ T_{\text{CONV}} = \text{Sampling time} + 12.5 \text{ ADC clock cycles} \]

Example:

With \( F_{\text{adc, ref, ck}} = 30 \text{ MHz} \) and a sampling time of 2.5 ADC clock cycles:

\[ T_{\text{CONV}} = (2.5 + 12.5) \text{ ADC clock cycles} = 15 \text{ ADC clock cycles} = 500 \text{ ns} \]

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

Note: Depending on the ADC conversion mode, the real sampling time can vary compared to the SMP value programmed above, while the equivalent total conversion time \( T_{\text{CONV}} \) does not change:

- For the first conversion in scan or continuous mode and all the conversions in discontinuous mode, the real sampling time is 0.5 clock cycle less compared to the value configured above.
- For the second and subsequent conversions in scan or continuous mode, 0.5 cycle is added to the configured sampling time. This additional 0.5 clock cycle overlaps with the previous conversion cycle.

Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC characteristics section of the datasheets.
Bulb sampling mode

When the BULB bit is set in ADC register, the sampling period starts immediately after the last ADC conversion. A hardware or software trigger starts the conversion after the sampling time has been programmed in ADC_SMPR1 register. The very first ADC conversion, after the ADC is enabled, is performed with the sampling time programmed in SMP bits. The bulb mode is effective starting from the second conversion.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

The bulb mode is neither compatible with the continuous conversion mode nor with the injected channel conversion.

When the BULB bit is set, it is not allowed to set SMPTRIG bit in ADC_CFGR2.

Sampling time control trigger mode

When the SMPTRIG bit is set, the sampling time programmed though SMPx bits is not applicable. The sampling time is controlled by the trigger signal edge.

When a hardware trigger is selected, each rising edge of the trigger signal starts the sampling period. A falling edge ends the sampling period and starts the conversion.

When a software trigger is selected, the software trigger is not the ADSTART bit in ADC_CR but the SWTRIG bit. SWTRIG bit has to be set to start the sampling period, and the SWTRIG bit has to be cleared to end the sampling period and start the conversion.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

This mode is neither compatible with the continuous conversion mode, nor with the injected channel conversion.

When SMPTRIG bit is set, it is not allowed to set BULB bit.

I/O analog switch voltage booster

The resistance of the I/O analog switches increases when the $V_{DDA}$ voltage is too low. The sampling time must consequently be adapted accordingly (refer to the device datasheet for the corresponding electrical characteristics). This resistance can be minimized at low $V_{DDA}$.
by enabling an internal voltage booster through BOOSTE bit or by selecting a $V_{DD}$ booster voltage (if $V_{DD} > 2.7$ V) through the ADVREADY bit of the PWR_PMCR register.

**SMPPLUS control bit**

The SMPPLUS bit can be used to change the sampling time from 2.5 ADC clock cycles to 3.5 ADC clock cycles.

### 26.4.13 Single conversion mode (CONT = 0)

In single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:

- Setting the ADSTART bit in the ADC_CR register (for a regular channel)
- Setting the JADSTART bit in the ADC_CR register (for an injected channel)
- External hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADC_DR register
- The EOC (end of regular conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 16-bit ADC_JDRy registers
- The JEOC (end of injected conversion) flag is set
- An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set
- An interrupt is generated if the JEOSIE bit is set

Then the ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

*Note: To convert a single channel, program a sequence with a length of 1.*

### 26.4.14 Continuous conversion mode (CONT = 1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts each conversions of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set
After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

**Note:**

*To convert a single channel, program a sequence with a length of 1.*

*It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.*

*Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to **Auto-injection mode** section.*

### 26.4.15 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART = 1.

When ADSTART is set, the conversion starts:

- Immediately: if EXTEN = 0x0 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTEN is not equal to 0x0

Software starts ADC injected conversions by setting JADSTART = 1.

When JADSTART is set, the conversion starts:

- Immediately, if JEXTEN = 0x0 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN is not equal to 0x0

**Note:**

*In auto-injection mode (JAUTO = 1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).*

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART = 0 and JADSTART = 0 are both true, indicating that the ADC is idle.

ADSTART is cleared by hardware:

- In single mode with software regular trigger (CONT = 0, EXTSEL = 0x0)
  - At any end of regular conversion sequence (EOS assertion) or at any end of subgroup processing if DISCEN = 1
- In all cases (CONT = x, EXTSEL = x)
  - After execution of the ADSTP procedure asserted by the software.

**Note:**

*In continuous mode (CONT = 1), ADSTART is not cleared by hardware with the assertion of EOS because the sequence is automatically relaunched.*

When a hardware trigger is selected in single mode (CONT = 0 and EXTSEL ≠ 0x00), ADSTART is not cleared by hardware with the assertion of EOS to help the software which does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.
JADSTART is cleared by hardware:

- In single mode with software injected trigger (JEXTSEL = 0x0)
  - At any end of injected conversion sequence (JEOS assertion) or at any end of subgroup processing if JDISCEN = 1
- In all cases (JEXTSEL = x)
  - After execution of the JADSTP procedure asserted by the software.

**Note:** When the software trigger is selected, ADSTART bit should not be set if the EOC flag is still high.

### 26.4.16 ADC timing

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

\[
T_{\text{CONV}} = T_{\text{SMPL}} + T_{\text{SAR}} = \left[ 2.5 \text{ min} + 12.5 \text{ 12bit} \right] \times T_{\text{ADC_CLK}}
\]

\[
T_{\text{CONV}} = T_{\text{SMPL}} + T_{\text{SAR}} = 83.33 \text{ ns}_{\text{min}} + 416.67 \text{ ns}_{\text{12bit}} = 500.0 \text{ ns (for } F_{\text{ADC_CLK}} = 30 \text{ MHz)}
\]

**Figure 196. Analog to digital conversion time**

1. \( T_{\text{SMPL}} \) depends on SMP[2:0].
2. \( T_{\text{SAR}} \) depends on RES[2:0].
26.4.17 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP = 1 and injected conversions ongoing by setting JADSTP = 1.

Stopping conversions resets the ongoing ADC operation. Then the ADC can be reconfigured (ex: changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADC_JDRy register is not updated with the current conversion).

The scan sequence is also aborted and reset (meaning that relaunching the ADC would restart a new sequence).

Once this procedure is complete, bits ADSTP/ADSTART (in case of regular conversion), or JADSTP/JADSTART (in case of injected conversion) are cleared by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

Note: In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

![Figure 197. Stopping ongoing regular conversions](MSv30533V2)
26.4.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (such as timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

When the Injected Queue is enabled (bit JQDIS = 0), injected software triggers are not possible.

The regular trigger selection is effective once software has set bit ADSTART = 1 and the injected trigger selection is effective once software has set bit JADSTART = 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

- If bit ADSTART = 0, any regular hardware triggers which occur are ignored.
- If bit JADSTART = 0, any injected hardware triggers which occur are ignored.

Table 243 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 243. Configuring the trigger polarity for regular external triggers

<table>
<thead>
<tr>
<th>EXTEN[1:0]</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Hardware Trigger detection disabled, software trigger detection enabled</td>
</tr>
<tr>
<td>01</td>
<td>Hardware Trigger with detection on the rising edge</td>
</tr>
<tr>
<td>10</td>
<td>Hardware Trigger with detection on the falling edge</td>
</tr>
<tr>
<td>11</td>
<td>Hardware Trigger with detection on both the rising and falling edges</td>
</tr>
</tbody>
</table>

Note: The polarity of the regular trigger cannot be changed on-the-fly.
Table 244. Configuring the trigger polarity for injected external triggers

<table>
<thead>
<tr>
<th>JEXTEN[1:0]</th>
<th>Source</th>
</tr>
</thead>
</table>
| 00          | - If JQDIS = 1 (Queue disabled): Hardware trigger detection disabled, software trigger detection enabled  
             | - If JQDIS = 0 (Queue enabled), Hardware and software trigger detection disabled                  |
| 01          | Hardware Trigger with detection on the rising edge                   |
| 10          | Hardware Trigger with detection on the falling edge                  |
| 11          | Hardware Trigger with detection on both the rising and falling edges |

Note: The polarity of the injected trigger can be anticipated and changed on-the-fly when the queue is enabled (JQDIS = 0). Refer to Section 26.4.21: Queue of context for injected conversions.

The EXTSEL and JEXTSEL control bits select which out of 32 possible events can trigger conversion for the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Note: The regular trigger selection cannot be changed on-the-fly. The injected trigger selection can be anticipated and changed on-the-fly. Refer to Section 26.4.21: Queue of context for injected conversions on page 1041.

Figure 199. Triggers shared between ADC master and slave
Refer to Table ADC interconnection in Section 26.4.2: ADC pins and internal signals for the list of all the external triggers that can be used for regular conversion.

26.4.19 Injected channel management

Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.

2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once).

3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.

4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. Figure 200 shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 30 ADC clock cycles (that is two conversions with a sampling time of 2.5 clock periods), the minimum interval between triggers must be 31 ADC clock cycles.

Auto-injection mode

If the JAUTO bit in the ADC_CFGR register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: It is not possible to use both the auto-injected and discontinuous modes simultaneously.

When the DMA is used for exporting regular sequencer’s data in JAUTO mode, it is necessary to program it in circular mode (CIRC bit set in DMA_CCRx register). If the CIRC bit is reset (single-shot mode), the JAUTO sequence is stopped upon DMA Transfer Complete event.
26.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN bit in the ADC_CFGR register.

It is used to convert a short sequence (subgroup) of n conversions \((n \leq 8)\) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of \(n\) is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR register.

When an external trigger occurs, it starts the next \(n\) conversions selected in the ADC_SQRy registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example:

- **DISCEN = 1, \(n = 3\)**, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
  - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
  - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - ... 
- **DISCEN = 0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11**
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events relaunch the complete sequence.

---

1. The maximum latency value can be found in the electrical characteristics of the device datasheet.
**Note:** The channel numbers referred to in the above example might not be available on all microcontrollers.

When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN = 1, CONT = 1), the ADC behaves as if continuous mode was disabled.

**Injected group mode**

This mode is enabled by setting the JDISCEN bit in the ADC_CFRGR register. It converts the sequence selected in the ADC_JSQ register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where 'n' is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQ registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQ register.

Example:

- JDISCEN = 1, channels to be converted = 1, 2, 3
  - 1st trigger: channel 1 converted (a JEOC event is generated)
  - 2nd trigger: channel 2 converted (a JEOC event is generated)
  - 3rd trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

**Note:** The channel numbers referred to in the above example might not be available on all microcontrollers.

When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

**26.4.21 Queue of context for injected conversions**

A queue of context is implemented to anticipate up to 2 contexts for the next injected sequence of conversions. JQDIS bit of ADC_CFRGR register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled.

This context consists of:

- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL bits in ADC_JSQ register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADC_JSQ register)
All the parameters of the context are defined into a single register ADC_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.
- A Queue overflow occurs when writing into register JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADC_CFGR:
  - If JQM = 0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence is served according to the last active context.
  - If JQM = 1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and hardware triggers are disabled. Therefore, any further hardware injected triggers are ignored until the software re-writes a new injected context into JSQR register.
- Reading JSQR register returns the current JSQR context which is active at that moment. When the JSQR context is empty, JSQR is read as 0x0000.
- The Queue is flushed when stopping injected conversions by setting JADSTP = 1 or when disabling the ADC by setting ADDIS = 1:
  - If JQM = 0, the Queue is maintained with the last active context.
  - If JQM = 1, the Queue becomes empty and triggers are ignored.

Note: When configured in discontinuous mode (bit JDISCEN = 1), only the last trigger of the injected sequence changes the context and consumes the Queue. The 1st trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):

- 1st trigger, discontinuous. Sequence 1: context 1 consumed, 1st conversion carried out
- 2nd trigger, disc. Sequence 1: 2nd conversion.
- 3rd trigger, discontinuous. Sequence 1: 3rd conversion.
- 4th trigger, discontinuous. Sequence 2: context 2 consumed, 1st conversion carried out.
- 5th trigger, discontinuous. Sequence 2: 2nd conversion.
- 6th trigger, discontinuous. Sequence 2: 3rd conversion.
Behavior when changing the trigger or sequence context

*Figure 201* and *Figure 202* show the behavior of the context Queue when changing the sequence or the triggers.

**Figure 201. Example of JSQR queue of context (sequence change)**

<table>
<thead>
<tr>
<th>Write JSQR</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSQR queue</td>
<td>EMPTY</td>
<td>P1</td>
<td>P1,P2</td>
</tr>
<tr>
<td>Trigger 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC J context (returned by reading JSQR)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC state</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Parameters:
P1: sequence of 3 conversions, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 4 conversions, hardware trigger 1

**Figure 202. Example of JSQR queue of context (trigger change)**

<table>
<thead>
<tr>
<th>Write JSQR</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSQR queue</td>
<td>EMPTY</td>
<td>P1</td>
<td>P1,P2</td>
</tr>
<tr>
<td>Trigger 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trigger 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC J context (returned by reading JSQR)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC state</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Parameters:
P1: sequence of 2 conversions, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 2
P3: sequence of 4 conversions, hardware trigger 1
Queue of context: Behavior when a queue overflow occurs

The Figure 203 and Figure 204 show the behavior of the context Queue if an overflow occurs before or during a conversion.

Figure 203. Example of JSQR queue of context with overflow before conversion

Figure 204. Example of JSQR queue of context with overflow during conversion

1. Parameters:
P1: sequence of 2 conversions, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 2
P3: sequence of 3 conversions, hardware trigger 1
P4: sequence of 4 conversions, hardware trigger 1
It is recommended to manage the queue overflows as described below:

- After each P context write into JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).
- Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

Queue of context: Behavior when the queue becomes empty

*Figure 205* and *Figure 206* show the behavior of the context Queue when the Queue becomes empty in both cases JQM = 0 or 1.

**Figure 205. Example of JSQR queue of context with empty queue (case JQM = 0)**

1. Parameters:
   - P1: sequence of 1 conversion, hardware trigger 1
   - P2: sequence of 1 conversion, hardware trigger 1
   - P3: sequence of 1 conversion, hardware trigger 1

   **Note:** When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.
Flushing the queue of context

The figures below show the behavior of the context Queue in various situations when the queue is flushed.

Figure 206. Example of JSQR queue of context with empty queue (JQM = 1)

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Figure 207. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion.

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1
Figure 208. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Figure 209. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1
Figure 210. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 1)

Queue is flushed and becomes empty (P2 is lost)

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Figure 211. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 0)

Queue is flushed and maintains the last active context (P2 which was not consumed is lost)

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1
Queue of context: Starting the ADC with an empty queue

The following procedure must be followed to start ADC operation with an empty queue, in case the first context is not known at the time the ADC is initialized. This procedure is only applicable when JQM bit is reset:

5. Write a dummy JSQR with JEXTEN not equal to 0 (otherwise triggering a software conversion)
6. Set JADSTART
7. Set JADSTP
8. Wait until JADSTART is reset
9. Set JADSTART.

Disabling the queue

It is possible to disable the queue by setting bit JQDIS = 1 into the ADC_CFGR register.

26.4.22 Programmable resolution (RES) - fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the control bits RES[1:0]. Figure 217, Figure 218, Figure 219 and Figure 220 show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to Table 245.
26.4.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application for each end of regular conversion (EOC) event and each injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADC_DR.

The ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADC_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

26.4.24 End of conversion sequence (EOS, JEOS)

The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADC_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

The ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.

<table>
<thead>
<tr>
<th>RES (bits)</th>
<th>$T_{SAR}$ (ADC clock cycles)</th>
<th>$T_{SAR}$ (ns) at $F_{ADC} = 30$ MHz</th>
<th>$T_{CONV}$ (ADC clock cycles) (with Sampling Time = 2.5 ADC clock cycles)</th>
<th>$T_{CONV}$ (ns) at $F_{ADC} = 30$ MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12.5 ADC clock cycles</td>
<td>416.67 ns</td>
<td>15 ADC clock cycles</td>
<td>500.0 ns</td>
</tr>
<tr>
<td>10</td>
<td>10.5 ADC clock cycles</td>
<td>350.0 ns</td>
<td>13 ADC clock cycles</td>
<td>433.33 ns</td>
</tr>
<tr>
<td>8</td>
<td>8.5 ADC clock cycles</td>
<td>203.33 ns</td>
<td>11 ADC clock cycles</td>
<td>366.67 ns</td>
</tr>
<tr>
<td>6</td>
<td>6.5 ADC clock cycles</td>
<td>216.67 ns</td>
<td>9 ADC clock cycles</td>
<td>300.0 ns</td>
</tr>
</tbody>
</table>
26.4.25 Timing diagrams example (single/continuous modes, hardware/software triggers)

**Figure 213. Single conversions of a sequence, software trigger**

1. EXTEN = 0x0, CONT = 0
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.

**Figure 214. Continuous conversion of a sequence, software trigger**

1. EXTEN = 0x0, CONT = 1
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.
Figure 215. Single conversions of a sequence, hardware trigger

1. TRGx (over-frequency) is selected as trigger source, EXTN = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

Figure 216. Continuous conversions of a sequence, hardware trigger

1. TRGx is selected as trigger source, EXTN = 10, CONT = 1
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.
26.4.26 Data management

Data register, data alignment and offset (ADC_DR, OFFSET, OFFSET_CH, ALIGN)

Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the converted data is stored into the ADC_DR data register which is 16 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the converted data is stored into the corresponding ADC_JDRy data register which is 16 bits wide.

The ALIGN bit in the ADC_CFGR register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in Figure 217, Figure 218, Figure 219 and Figure 220.

Special case: when left-aligned, the data are aligned on a half-word basis except when the resolution is set to 6-bit. In that case, the data are aligned on a byte basis as shown in Figure 219 and Figure 220.

Note: Left-alignment is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the ALIGN bit value is ignored and the ADC only provides right-aligned data.

Offset

An offset y (y = 1,2,3,4) can be applied to a channel by setting the bit OFFSET_EN = 1 into ADC_OFRy register. The channel to which the offset will be applied is programmed into the bits OFFSET_CH[4:0] of ADC_OFRy register. In this case, the converted value is decreased by the user-defined offset written in the bits OFFSET[11:0]. The result may be a negative value so the read data is signed and the SEXT bit represents the extended sign value.

Note: Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSET_EN bit in ADC_OFRy register is ignored (considered as reset).

Table 248 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

<table>
<thead>
<tr>
<th>Resolution (bits RES[1:0])</th>
<th>Subtraction between raw converted data and offset</th>
<th>Result</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Raw converted Data, left aligned</td>
<td>Offset</td>
<td></td>
</tr>
<tr>
<td>00: 12-bit</td>
<td>DATA[11:0] OFFSET[11:0]</td>
<td>Signed 12-bit data</td>
<td>-</td>
</tr>
<tr>
<td>01: 10-bit</td>
<td>DATA[11:2],00 OFFSET[11:0]</td>
<td>Signed 10-bit data</td>
<td>The user must configure OFFSET[1:0] to &quot;00&quot;</td>
</tr>
</tbody>
</table>
When reading data from ADC_DR (regular channel) or from ADC_JDRy (injected channel, y = 1,2,3,4) corresponding to the channel “i”:
- If one of the offsets is enabled (bit OFFSET_EN = 1) for the corresponding channel, the read data is signed.
- If none of the four offsets is enabled for this channel, the read data is not signed.

*Figure 217, Figure 218, Figure 219* and *Figure 220* show alignments for signed and unsigned data.

### Table 246. Offset computation versus data resolution (continued)

<table>
<thead>
<tr>
<th>Resolution (bits RES[1:0])</th>
<th>Subtraction between raw converted data and offset</th>
<th>Result</th>
<th>Comments</th>
</tr>
</thead>
</table>

*Figure 217. Right alignment (offset disabled, unsigned value)*

<table>
<thead>
<tr>
<th>12-bit data</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>10-bit data</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8-bit data</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0 D7 D6 D5 D4 D3 D2 D1 D0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6-bit data</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0 D5 D4 D3 D2 D1 D0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Figure 218. Right alignment (offset enabled, signed value)

<table>
<thead>
<tr>
<th>12-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEXT SEXT SEXT SEXT D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>10-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEXT SEXT SEXT SEXT D9 D8 D7 D6 D5 D4 D3 D2 D1 D0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEXT SEXT SEXT SEXT D7 D6 D5 D4 D3 D2 D1 D0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEXT SEXT SEXT SEXT SEXT SEXT D5 D4 D3 D2 D1 D0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Figure 219. Left alignment (offset disabled, unsigned value)

<table>
<thead>
<tr>
<th>12-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>10-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7 D6 D5 D4 D3 D2 D1 D0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0 D5 D4 D3 D2 D1 D0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Offset compensation

When SATEN bit is set in ADC_OFRy register during offset operation, data are unsigned. All the offset data saturate at 0x000 (in 12-bit mode). When OFFSETPOS bit is set, the offset direction is positive and the data saturate at 0xFFF (in 12-bit mode). In 8-bit mode, data saturate at 0x00 and 0xFF, respectively.

The analog watchdog comparison is performed before the offset compensation.

ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) notifies when the regular converted data has not been read (by the CPU or the DMA) before ADC_DR FIFO (three stages) is overflowed.

The OVR flag is set when a new conversion completes while ADC_CR register FIFO was full. An interrupt is generated if OVRIE bit is set to 1.

When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting ADSTP to 1. Since ADC_DR FIFO features three stages, up to three data are stored in the FIFO.

OVR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- **OVRMOD = 0**: The overrun event preserves the data register from being overwritten: the old data is maintained up to ADC_DR FIFO depth (three stages) and the new conversion is discarded and lost. In this mode, ADC_DR FIFO is disabled. If the FIFO is full, any further conversion is performed but the resulting data is also discarded. EOC
is cleared by reading ADC_DR register. However, the FIFO can still contain previously converted data.

- OVRMOD = 1: The data register is overwritten with the last conversion result and the previous unread data is lost. In this mode, ADC_DR FIFO is disabled. If OVR remains at 1, any further conversions is performed normally and the ADC_DR register always contains the latest converted data.

**Figure 221. Example of overrun (OVRMOD = 0)**
Note: There is no overrun detection on the injected channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD must be configured to 0 to manage overrun events or FIFO overflow as an error.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVR flag should be ignored by the software. An overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion.

Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMAEN bit set to 1 in the ADC_CFGR register in single ADC mode or MDMA different from 0b00 in dual ADC mode), a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.
Despite this, if an overrun occurs (OVR = 1) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to Section: ADC overrun (OVR, OVRMOD)).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMACFG of the ADC_CFGR register in single ADC mode, or with bit DMACFG of the ADC_CCR register in dual ADC mode:

- DMA one shot mode (DMACFG = 0)
  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMACFG = 1)
  This mode is suitable when programming the DMA in circular mode.

DMA one shot mode (DMACFG = 0)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when a transfer complete interrupt occurs - refer to DMA section) even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted with partial result discarded.
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- Scan sequence is stopped and reset.
- The DMA is stopped.

DMA circular mode (DMACFG = 1)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

26.4.27 Dynamic low-power features

Auto-delayed conversion mode (AUTDLY)

The ADC implements an auto-delayed conversion mode controlled by the AUTDLY configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.
When AUTDLY = 1, a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the ADC_DR register has been read or if the EOC bit has been cleared (see Figure 223).
- For an injected conversion: when the JEOS bit has been cleared (see Figure 224).

This is a way to automatically adapt the speed of the ADC to the speed of the system which reads the data.

The delay is inserted after each regular conversion (whatever DISCEN = 0 or 1) and after each sequence of injected conversions (whatever JDISCEN = 0 or 1).

**Note:** There is no delay inserted between each conversions of the injected sequence, except after the last one.

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

**Note:** This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to restart a conversion: it is up to the software to read the data before launching a new conversion.

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see Figure 224).
- Once the injected sequence is complete, the ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see Figure 226).

The behavior is slightly different in auto-injected mode (JAUTO = 1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see Figure 227).

To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO = 1, CONT = 1 and AUTDLY = 1), follow the following procedure:

1. Wait until JEOS = 1 (no more conversions are restarted)
2. Clear JEOS,
3. Set ADSTP = 1
4. Read the regular data.

If this procedure is not respected, a new regular sequence can restart if JEOS is cleared after ADSTP has been set.

In AUTDLY mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.
Figure 223. AUTODLY = 1, regular conversion in continuous mode, software trigger

1. AUTDLY = 1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT = 1, CHANNELS = 1,2,3
3. Injected configuration DISABLED

Figure 224. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)

1. AUTDLY = 1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT = 0, DISCEN = 0, CHANNELS = 1,2,3
3. Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 0, CHANNELS = 5,6
Figure 225. AUTDLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1)

1. AUTDLY = 1
2. Regular configuration: EXTEN = 0x1 (HW trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 1, CHANNELS = 5, 6
Figure 226. AUTODLY = 1, regular continuous conversions interrupted by injected conversions

1. AUTDLY = 1
2. Regular configuration: EXTEN = 0x0 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 0, CHANNELS = 5, 6

Figure 227. AUTODLY = 1 in auto- injected mode (JAUTO = 1)

1. AUTDLY = 1
2. Regular configuration: EXTEN = 0x0 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2
3. Injected configuration: JAUTO = 1, CHANNELS = 5, 6
26.4.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

**Figure 228. Analog watchdog guarded area**

AWDx flag and interrupt

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDxIE in the ADC_IER register (x = 1,2,3).

AWDx (x = 1,2,3) flag is cleared by software by writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

**Description of analog watchdog 1**

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADC_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels\(^{(1)}\) remain within a configured voltage range (window).

*Table 247* shows how the ADC_CFGR registers should be configured to enable the analog watchdog on one or more channels.

**Table 247. Analog watchdog channel selection**

<table>
<thead>
<tr>
<th>Channels guarded by the analog watchdog</th>
<th>AWD1SGL bit</th>
<th>AWD1EN bit</th>
<th>JAWD1EN bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>All injected channels</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>All regular channels</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>All regular and injected channels</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Single(^{(1)}) injected channel</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Single(^{(1)}) regular channel</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Single(^{(1)}) regular or injected channel</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. Selected by the AWD1[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.
These thresholds are programmed in bits HT1[11:0] and LT1[11:0] of the ADC_TR1 register for the analog watchdog 1. When converting data with a resolution of less than 12 bits (according to bits RES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned) before the offset compensation stage.

*Table 248* describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

### Table 248. Analog watchdog 1 comparison

<table>
<thead>
<tr>
<th>Resolution (bit RES[1:0])</th>
<th>Analog watchdog comparison between:</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>00: 12-bit DATA[11:0]</td>
<td>LT1[11:0] and HT1[11:0]</td>
<td>-</td>
</tr>
<tr>
<td>01: 10-bit DATA[11:2],00</td>
<td>LT1[11:0] and HT1[11:0]</td>
<td>User must configure LT1[1:0] and HT1[1:0] to 00</td>
</tr>
</tbody>
</table>

### Analog watchdog filter for watchdog 1

When an ADC is configured with only one input channel (selecting several channels in scan mode not allowed), a valid ADC conversion data interval can be configured through the ADC_TR1 register:

- When converted data belong to the interval defined in ADC_TR1, a DMA request is generated.
- Otherwise, no DMA request is issued. RDATA register is updated at each conversion. If data are out-of-range a number of times higher than the value specified in AWDFILT bit of ADC_TR1, the AWDx flag is set an the corresponding interrupt is issued.

### Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDxCH[19:0] (x = 2,3). The corresponding watchdog is enabled when any bit of AWDxCH[19:0] (x = 2,3) is set.

They are limited to a resolution of 8 bits and only the 8 MSBs of the thresholds can be programmed into HTx[7:0] and LTx[7:0]. *Table 249* describes how the comparison is performed for all the possible resolutions.
ADCy_AWDx_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal ADCy_AWDx_OUT (y = ADC number, x = watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADCy_AWDx_OUT signal as ETR.

ADCy_AWDx_OUT is activated when the associated analog watchdog is enabled:

- ADCy_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADCy_AWDx_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADCy_AWDx_OUT is also reset when disabling the ADC (when setting ADDIS = 1).

Note: AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADCy_AWDx_OUT (ex: ADCy_AWDx_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag).

Table 249. Analog watchdog 2 and 3 comparison

<table>
<thead>
<tr>
<th>Resolution (bits RES[1:0])</th>
<th>Analog watchdog comparison between:</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>00: 12-bit</td>
<td>DATA[11:4] LTx[7:0] and HTx[7:0] DATA[3:0] are not relevant for the comparison</td>
<td>ADCy_AWDx_OUT inside cleared by S/W</td>
</tr>
<tr>
<td>11: 6-bit</td>
<td>DATA[11:6],00 LTx[7:0] and HTx[7:0] User must configure LTx[1:0] and HTx[1:0] to 00</td>
<td>ADCy_AWDx_OUT inside cleared by S/W</td>
</tr>
</tbody>
</table>

Figure 229. ADCy_AWDx_OUT signal generation (on all regular channels)
Figure 230. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software)

- Converting regular channels 1,2,3,4,5,6,7
- Regular channels 1,2,3,4,5,6,7 are all guarded

Figure 231. ADCy_AWDx_OUT signal generation (on a single regular channel)

- Converting regular channels 1 and 2
- Only channel 1 is guarded

Figure 232. ADCy_AWDx_OUT signal generation (on all injected channels)

- Converting the injected channels 1, 2, 3, 4
- All injected channels 1, 2, 3, 4 are guarded
Analog watchdog threshold control

LTx[11:0] and HTx[11:0] can be changed when an analog-to-digital conversion is ongoing (that is between the start of conversion and the end of conversion of the ADC internal state). If LTx[11:0] and HTx[11:0] are updated during the ADC conversion of the ADC guarded channel, the watchdog function is masked for this conversion. This masking is removed at the next start of conversion, resulting in analog watchdog thresholds to be applied from the next ADC conversion. The analog watchdog comparison is performed at each end of conversion. If the current ADC data is out of the new interval, no interrupt and AWDx_OUT signal are issued. The Interrupt and the AWD generation only happen at the end of the conversion which started after the threshold update. If AWD_xOUT is already asserted, programming the new thresholds does not deassert the AWDx_OUT signal.

Analog watchdog with offset compensation

When the offset compensation is enabled, the analog watchdog compares the threshold before the data compensation.

26.4.29 Oversampler

The oversampling unit performs data pre-processing to offload the CPU. It is able to handle multiple conversions and average them into a single data with increased data width, up to 16-bit.

It provides a result with the following form, where N and M can be adjusted:

\[
\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)
\]

It allows to perform by hardware the following functions: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVFS[2:0] bits in the ADC_CFGR2 register, and can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits, and is defined using the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 20 bits (256x 12-bit results), which is first shifted right. It is then truncated to the 16 least significant bits, rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the ADC_DR data register.

Note: If the intermediary result after the shifting exceeds 16-bit, the result is truncated as is, without saturation.
Figure 233. 20-bit to 16-bit result truncation

Figure 234 gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

Figure 234. Numerical example with 5-bit shift and rounding

Table 250 gives the data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.

Table 250. Maximum output results versus N and M (gray cells indicate truncation)

<table>
<thead>
<tr>
<th>Over sampling ratio</th>
<th>Max Raw data</th>
<th>No-shift</th>
<th>1-bit shift</th>
<th>2-bit shift</th>
<th>3-bit shift</th>
<th>4-bit shift</th>
<th>5-bit shift</th>
<th>6-bit shift</th>
<th>7-bit shift</th>
<th>8-bit shift</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>OVSS = 0000</td>
<td>OVSS = 0001</td>
<td>OVSS = 0010</td>
<td>OVSS = 0011</td>
<td>OVSS = 0100</td>
<td>OVSS = 0101</td>
<td>OVSS = 0110</td>
<td>OVSS = 0111</td>
<td>OVSS = 1000</td>
</tr>
<tr>
<td>2x</td>
<td>0x1FFE</td>
<td>0x1FFE</td>
<td>0x0FFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0100</td>
<td>0x0080</td>
<td>0x0040</td>
<td>0x0020</td>
<td>0x0010</td>
</tr>
<tr>
<td>4x</td>
<td>0x3FFC</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0FFFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0100</td>
<td>0x0080</td>
<td>0x0040</td>
<td>0x0020</td>
</tr>
<tr>
<td>8x</td>
<td>0x7FF8</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0100</td>
<td>0x0080</td>
<td>0x0040</td>
<td>0x0020</td>
</tr>
<tr>
<td>16x</td>
<td>0xFFFF0</td>
<td>0xFFFF0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0FFFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
</tr>
<tr>
<td>32x</td>
<td>0x1FFE0</td>
<td>0x1FFE0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFFF</td>
<td>0x0FFFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
</tr>
<tr>
<td>64x</td>
<td>0x3FFC0</td>
<td>0x3FFC0</td>
<td>0xFFFF0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0FFFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
</tr>
<tr>
<td>128x</td>
<td>0x7FF80</td>
<td>0x7FF80</td>
<td>0xFFFF0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0FFFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
</tr>
<tr>
<td>256x</td>
<td>0xFFFF00</td>
<td>0xFFFF00</td>
<td>0xFFFF0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0FFFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
</tr>
</tbody>
</table>

There are no changes for conversion timings in oversampled mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N
conversions, with an equivalent delay equal to $N \times T_{\text{CONV}} = N \times (t_{\text{SMPL}} + t_{\text{SAR}})$. The flags are set as follow:

- The end of the sampling phase (EOSMP) is set after each sampling phase
- The end of conversion (EOC) occurs once every $N$ conversions, when the oversampled result is available
- The end of sequence (EOS) occurs once the sequence of oversampled data is completed (that is after $N \times$ sequence length conversions total)

**ADC operating modes supported when oversampling (single ADC mode)**

In oversampling mode, most of the ADC operating modes are maintained:

- Single or continuous conversion modes
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (AUTDLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits in ADC_CFGR register) are accumulated, truncated, rounded and shifted in the same way as 12-bit conversions are

**Note:** The alignment mode is not available when working with oversampled data. The ALIGN bit in ADC_CFGR is ignored and the data are always provided right-aligned.

Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSET_EN bit in ADC_OFRy register is ignored (considered as reset).

**Analog watchdog**

The analog watchdog functionality is maintained, with the following difference:

- The RES[1:0] bits are ignored, the comparison is always done by using the full 12-bit values HT1[11:0] and LT1[11:0] for AWD1, and the 8-MSB value of HT2/HT3[7:0] and LT2/LT3[7:0] for AWD2 and AWD3.
- The comparison is done on the most significant 12-bit of the 16-bit oversampled results ADC_DR[15:4] for AWD1, and ADC_DR[15:8] for AWD2 and ADW3.

**Note:** Care must be taken when using high shifting values, this reduces the comparison range. For instance, if the oversampled result is shifted by 4 bits, thus yielding a 12-bit data right-aligned, the effective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC_DR[11:4] and HTx[7:0] / LTx[7:0] (AWD1/2/3), with HT1[11:8] and LT1[11:8] kept reset (AWD1 only).

**Triggered mode**

The averager can also be used for basic filtering purpose. Although not a very powerful filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific discontinuous mode can be enabled with TROVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

The Figure 235 below shows how conversions are started in response to triggers during discontinuous mode.
If the TROVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

**Figure 235. Triggered regular oversampling mode (TROVS bit = 1)**

<table>
<thead>
<tr>
<th>CONT=0</th>
<th>Trigger</th>
<th>Trigger</th>
<th>Trigger</th>
<th>Trigger</th>
<th>Trigger</th>
<th>Trigger</th>
<th>Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISCEN = 1</td>
<td>Ch(N)3</td>
<td>Ch(N)2</td>
<td>Ch(N)1</td>
<td>Ch(N)0</td>
<td>Ch(N)3</td>
<td>Ch(N)2</td>
<td>Ch(N)1</td>
</tr>
<tr>
<td>TROVS = 0</td>
<td>EOC flag set</td>
<td>EOC flag set</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Injected and regular sequencer management when oversampling**

In oversampling mode, it is possible to have differentiated behavior for injected and regular sequencers. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

**Oversampling regular channels only**

The regular oversampling mode bit ROVSM defines how the regular oversampling sequence is resumed if it is interrupted by injected conversion:

- In continued mode, the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling is complete whatever the injection frequency (providing at least one regular conversion can be completed between triggers);
- In resumed mode, the accumulation restarts from 0 (previous conversions results are ignored). This mode allows to guarantee that all data used for oversampling were converted back-to-back within a single timeslot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be completed and the regular sequencer is blocked.

The **Figure 236** gives examples for a 4x oversampling ratio.
**Figure 236. Regular oversampling modes (4x ratio)**

Oversampling Injected channels only

The Injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.
Oversampling regular and injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling mode is forced to resumed mode (ROVSM bit ignored), as represented on Figure 237 below.

Figure 237. Regular and injected oversampling modes used simultaneously

Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected mode oversampling mode must be disabled, and the ROVSM bit is ignored (resumed mode is forced). The JOVSE bit must be reset. The behavior is represented on Figure 238 below.
Auto-injected mode

It is possible to oversample auto-injected sequences and have all conversions results stored in registers to save a DMA resource. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1, other combinations are not supported. The ROVSM bit is ignored in auto-injected mode. The Figure 239 below shows how the conversions are sequenced.

![Figure 239. Oversampling in auto-injected mode](image)

It is possible to have also the triggered mode enabled, using the TROVS bit. In this case, the ADC must be configured as following: JAUTO = 1, DISCEN = 0, JDISCEN = 0, ROVSE = 1, JOVSE = 1 and TROVSE = 1.

Dual ADC modes supported when oversampling

It is possible to have oversampling enabled when working in dual ADC configuration, for the injected simultaneous mode and regular simultaneous mode. In this case, the two ADCs must be programmed with the very same settings (including oversampling).

All other dual ADC modes are not supported when either regular or injected oversampling is enabled (ROVSE = 1 or JOVSE = 1).

Combined modes summary

The Table 251 below summarizes all combinations, including modes not supported.

<table>
<thead>
<tr>
<th>Regular Oversampling ROVSE</th>
<th>Injected Oversampling JOVSE</th>
<th>Oversampler mode ROVSM</th>
<th>Triggered Regular mode TROVS</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Regular continued mode</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Not supported</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Regular resumed mode</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Triggered regular resumed mode</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>Not supported</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Injected and regular resumed mode</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Not supported</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Injected oversampling</td>
</tr>
</tbody>
</table>
26.4.30 Dual ADC modes

Dual ADC modes can be used in devices with two ADCs or more (see Figure 240).

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADCx master to the ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADC_CCR register.

Four possible modes are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:

- Injected simultaneous mode + regular simultaneous mode
- Regular simultaneous mode + alternate trigger mode
- Injected simultaneous mode + interleaved mode

In dual ADC mode (when bits DUAL[4:0] in ADC_CCR register are not equal to zero), the bits CONT, AUTDLY, DISCEN, DISCNUM[2:0], JDISCE, JQM, JAUTO of the ADC_CFGR register are shared between the master and slave ADC: the bits in the slave ADC are always equal to the corresponding bits of the master ADC.

To start a conversion in dual mode, the user must program the bits EXTEN, EXTSEL, JEXTEN, JEXTSEL of the master ADC only, to configure a software or hardware trigger, and a regular or injected trigger. (the bits EXTEN[1:0] and JEXTEN[1:0] of the slave ADC are don’t care).

In regular simultaneous or interleaved modes: once the user sets bit ADSTART or bit ADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit ADSTART or bit ADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In injected simultaneous or alternate trigger modes: once the user sets bit JADSTART or bit JADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit JADSTART or bit JADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADC_CDR). The status bits can be also read in parallel by reading the dual-mode status register (ADC_CSR).
1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.

2. The ADC common data register (ADC_CDR) contains both the master and slave ADC regular converted data.
Injected simultaneous mode

This mode is selected by programming bits DUAL[4:0] = 00101

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL bits in the ADC_JSQR register).

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longer of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

- At the end of injected sequence of conversion event (JEOS) on the master ADC, the converted data is stored into the master ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- At the end of injected sequence of conversion event (JEOS) on the slave ADC, the converted data is stored into the slave ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- If the duration of the master injected sequence is equal to the duration of the slave injected one (like in Figure 241), it is possible for the software to enable only one of the two JEOS interrupt (ex: master JEOS) and read both converted data (from master ADC_JDRy and slave ADC_JDRy registers).

Figure 241. Injected simultaneous mode on 4 channels: dual ADC mode

If JDISCEN = 1, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADC_DR) has been read. Any new regular trigger events occurring for the master ADC during the
Analog-to-digital converters (ADC1/2) RM0481

ongoing regular sequence and the associated delay phases are ignored. There is the same behavior for regular sequences occurring on the slave ADC.

**Regular simultaneous mode with independent injected**

This mode is selected by programming bits DUAL[4:0] = 00110.

This mode is performed on a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL bits in the ADC_CFGR register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on master or on the slave) aborts the current simultaneous conversions, which are restarted once the injected conversion is completed.

*Note:* *Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).*

In regular simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longer conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Software is notified by interrupts when it can read the data:

- At the end of each conversion event (EOC) on the master ADC, a master EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the master ADC.
- At the end of each conversion event (EOC) on the slave ADC, a slave EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the slave ADC.
- If the duration of the master regular sequence is equal to the duration of the slave one (like in Figure 242), it is possible for the software to enable only one of the two EOC interrupt (ex: master EOC) and read both converted data from the Common Data register (ADC_CDR).

It is also possible to read the regular data using the DMA. Two methods are possible:

- Using two DMA channels (one for the master and one for the slave). In this case bits MDMA[1:0] must be kept cleared.
  - Configure the DMA master ADC channel to read ADC_DR from the master. DMA requests are generated at each EOC event of the master ADC.
  - Configure the DMA slave ADC channel to read ADC_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Using MDMA mode, which leaves one DMA channel free for other uses:
  - Configure MDMA[1:0] = 0b10 or 0b11 (depending on resolution).
  - A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADC_CDR)
  - A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADC_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADC_CDR register.
  - Both EOC flags are cleared when the DMA reads the ADC_CDR register.

*Note:* *In MDMA mode (MDMA[1:0] = 0b10 or 0b11), the user must program the same number of conversions in the master’s sequence as in the slave’s sequence. Otherwise, the remaining conversions does not generate a DMA request.*
If DISCEN = 1 then each "n" simultaneous conversions of the regular sequence require a regular trigger event to occur ("n" is defined by DISCNUM).

This mode can be combined with AUTDLY mode:

- Once a simultaneous conversion of the sequence has ended, the next conversion in the sequence is started only if the common data register, ADC_CDR (or the regular data register of the master ADC) has been read (delay phase).
- Once a simultaneous regular sequence of conversions has ended, a new regular trigger event is accepted only if the common data register (ADC_CDR) has been read (delay phase). Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

It is possible to use the DMA to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multiple-DMA mode is used: bits MDMA must be set to 0b10 or 0b11.

When regular simultaneous mode is combined with AUTDLY mode, it is mandatory for the user to ensure that:

- The number of conversions in the master’s sequence is equal to the number of conversions in the slave’s.
- For each simultaneous conversions of the sequence, the length of the conversion of the slave ADC is inferior to the length of the conversion of the master ADC. Note that the length of the sequence depends on the number of channels to convert and the sampling time and the resolution of each channels.

**Note:** This combination of regular simultaneous mode and AUTDLY mode is restricted to the use case when only regular channels are programmed: it is forbidden to program injected channels in this combined mode.

**Interleaved mode with independent injected**

This mode is selected by programming bits DUAL[4:0] = 00111.

This mode can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately.
- The slave ADC starts after a delay of several-ADC clock cycles after the sampling phase of the master ADC has complete.

The minimum delay which separates two conversions in interleaved mode is configured in the DELAY bits in the ADC_CCR register. This delay starts counting one half cycle after the end of the sampling phase of the master conversion. This way, an ADC cannot start a
conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

If the CONT bit is set on both master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

The software is notified by interrupts when it can read the data at the end of each conversion event (EOC) on the slave ADC. A slave and master EOC interrupts are generated (if EOCIE is enabled) and the software can read the ADC_DR of the slave/master ADC.

**Note:** It is possible to enable only the EOC interrupt of the slave and read the common data register (ADC_CDR). But in this case, the user must ensure that the duration of the conversions are compatible to ensure that inside the sequence, a master conversion is always followed by a slave conversion before a new master conversion restarts. It is recommended to use the MDMA mode.

It is also possible to have the regular data transferred by DMA. In this case, individual DMA requests on each ADC cannot be used and it is mandatory to use the MDMA mode, as following:

- Configure MDMA[1:0] = 0b10 or 0b11 (depending on resolution).
- A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADC_CDR).
- A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADC_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADC_CCR register.
- Both EOC flags are cleared when the DMA reads the ADC_CCR register.

**Figure 243. Interleaved mode on one channel in continuous conversion mode: dual ADC mode**

![Diagram](https://example.com/diagram.png)
If DISCEN = 1, each “n” simultaneous conversions (“n” is defined by DISCNUM) of the regular sequence require a regular trigger event to occur.

In this mode, injected conversions are supported. When injection is done (either on master or on slave), both the master and the slave regular conversions are aborted and the sequence is restarted from the master (see Figure 245 below).

**Alternate trigger mode**

This mode is selected by programming bits DUAL[4:0] = 01001.

This mode can be started only on an injected group. The source of external trigger comes from the injected group multiplexer of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN must not be 0x0.
Injected discontinuous mode disabled (JDISCEN = 0 for both ADC)

1. When the 1st trigger occurs, all injected master ADC channels in the group are converted.
2. When the 2nd trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on.

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts by converting the injected channels of the master ADC in the group.

**Figure 246. Alternate trigger: injected group of each ADC**

**Note:** Regular conversions can be enabled on one or all ADCs. In this case the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between 2 trigger events must be greater than or equal to 1 ADC clock period. The minimum time interval between 2 trigger events that start conversions on the same ADC is the same as in the single ADC mode.
Injected discontinuous mode enabled (JDISCEN = 1 for both ADC)

If the injected discontinuous mode is enabled for both master and slave ADCs:

- When the 1st trigger occurs, the first injected channel of the master ADC is converted.
- When the 2nd trigger occurs, the first injected channel of the slave ADC is converted.
- And so on.

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversions.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts.

**Figure 247. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode**

Combined regular/injected simultaneous mode

This mode is selected by programming bits DUAL[4:0] = 00001.

It is possible to interrupt the simultaneous conversion of a regular group to start the simultaneous conversion of an injected group.

*Note:* In combined regular/injected simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Combined regular simultaneous + alternate trigger mode

This mode is selected by programming bits DUAL[4:0] = 00010.

It is possible to interrupt the simultaneous conversion of a regular group to start the alternate trigger conversion of an injected group. **Figure 248** shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion is immediately started after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.
Note: In combined regular simultaneous + alternate trigger mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

**Figure 248. Alternate + regular simultaneous**

If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. **Figure 249** shows the behavior in this case (note that the 6th trigger is ignored because the associated alternate conversion is not complete).

**Figure 249. Case of trigger occurring during injected conversion**

**Combined injected simultaneous plus interleaved**

This mode is selected by programming bits DUAL[4:0] = 00011

It is possible to interrupt an interleaved conversion with a simultaneous injected event.

In this case the interleaved conversion is interrupted immediately and the simultaneous injected conversion starts. At the end of the injected sequence the interleaved conversion is resumed. When the interleaved regular conversion resumes, the first regular conversion which is performed is always the master’s one. **Figure 250, Figure 251** and **Figure 252** show the behavior using an example.

**Caution:** In this mode, it is mandatory to use the Common Data Register to read the regular data with a single read access. On the contrary, master-slave data coherency is not guaranteed.
Figure 250. Interleaved single channel CH0 with injected sequence CH11, CH12

![Interleaved single channel CH0 with injected sequence CH11, CH12](image)

Legend:

- Sampling
- Conversion

Injected trigger

Conversions aborted

Resume (always restart with the master)

---

Figure 251. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12
- case 1: Master interrupted first

![Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12](image)

Legend:

- Sampling
- Conversion

Injected trigger

Conversions aborted

Resume (always restart with the master)

---

Figure 252. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12
- case 2: Slave interrupted first

![Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12](image)

Legend:

- Sampling
- Conversion

Injected trigger

Conversions aborted

Resume (always restart with the master)
DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to Figure 253: DMA Requests in regular simultaneous mode when MDMA = 0b00).

Figure 253. DMA Requests in regular simultaneous mode when MDMA = 0b00

In simultaneous regular and interleaved modes, it is also possible to save one DMA channel and transfer both data using a single DMA channel. For this MDMA bits must be configured in the ADC_CCR register:

- **MDMA = 0b10**: A single DMA request is generated each time both master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADC_CDR contains the two half-words representing two ADC-converted data items. The slave ADC data take the upper half-word and the master ADC data take the lower half-word.
  
  This mode is used in interleaved mode and in regular simultaneous mode when resolution is 10-bit or 12-bit.

**Example:**

Interleaved dual mode: a DMA request is generated each time 2 data items are available:

1st DMA request: \( \text{ADC}_\text{CDR}[31:0] = \text{SLV}_\text{ADC}_\text{DR}[15:0] | \text{MST}_\text{ADC}_\text{DR}[15:0] \)

2nd DMA request: \( \text{ADC}_\text{CDR}[31:0] = \text{SLV}_\text{ADC}_\text{DR}[15:0] | \text{MST}_\text{ADC}_\text{DR}[15:0] \)
Figure 254. DMA requests in regular simultaneous mode when MDMA = 0b10

Figure 255. DMA requests in interleaved mode when MDMA = 0b10
Note: When using MDMA mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion is available.

- MDMA = 0b11: This mode is similar to the MDMA = 0b10. The only differences are that on each DMA request (two data items are available), two bytes representing two ADC converted data items are transferred as a half-word.
  This mode is used in interleaved and regular simultaneous mode when resolution is 6-bit or when resolution is 8-bit and data is not signed (offsets must be disabled for all the involved channels).

  **Example:**

  Interleaved dual mode: a DMA request is generated each time 2 data items are available:

  1st DMA request: ADC_CDR[15:0] = SLV_ADC_DR[7:0] | MST_ADC_DR[7:0]
  2nd DMA request: ADC_CDR[15:0] = SLV_ADC_DR[7:0] | MST_ADC_DR[7:0]

**Overrun detection**

In dual ADC mode (when DUAL[4:0] is not equal to b00000), if an overrun is detected on one of the ADCs, the DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the MDMA configuration). It may happen that the EOC bit corresponding to one ADC remains set because the data register of this ADC contains valid data.

**DMA one shot mode/ DMA circular mode when MDMA mode is selected**

When MDMA mode is selected (0b10 or 0b11), bit DMACFG of the ADC_CCR register must also be configured to select between DMA one shot mode and circular mode, as explained in section **Section : Managing conversions using the DMA** (bits DMACFG of master and slave ADC_CFGR are not relevant).

**Stopping the conversions in dual ADC modes**

The user must set the control bits ADSTP/JADSTP of the master ADC to stop the conversions of both ADC in dual ADC mode. The other ADSTP control bit of the slave ADC has no effect in dual ADC mode.

Once both ADC are effectively stopped, the bits ADSTART/JADSTART of the master and slave ADCs are both cleared by hardware.

**26.4.31 Temperature sensor**

The temperature sensor can be used to measure the junction temperature (Tj) of the device.

The temperature sensor is internally connected to the ADC input channels which are used to convert the sensor output voltage to a digital value (see **Table: ADC interconnection** in **Section 26.4.2: ADC pins and internal signals** for more details). When not in use, the sensor can be put in power down mode. It support the temperature range –40 to 125 °C.

**Figure 256** shows the block diagram of connections between the temperature sensor and the ADC.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).
The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference (refer to the datasheet for additional information).

The temperature sensor is internally connected to the ADC input channel which is used to convert the sensor’s output voltage to a digital value. Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal temperature sensor.

When not in use, the sensor can be put in power-down mode. 

*Figure 256* shows the block diagram of the temperature sensor.

---

**Figure 256. Temperature sensor channel block diagram**

---

**Reading the temperature**

To use the sensor:

1. Select the ADC input channels that is connected to $V_{SENSE}$.
2. Program with the appropriate sampling time (refer to electrical characteristics section of the device datasheet).
3. Set the bit in the ADC_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting $V_{SENSE}$ data in the ADC data register.
6. Calculate the actual temperature using the following formula:
where:
- TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP.
- TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP.
- TS_DATA is the actual temperature sensor output value converted by ADC.

Refer to the device datasheet for more information about TS_CAL1 and TS_CAL2 calibration points.

**Note:** The sensor has a startup time after waking from power-down mode before it can output VSENSE at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and bits should be set at the same time.

### 26.4.32 V\textsubscript{BAT} supply monitoring

The VBATEN bit in the ADC_CCR register is used to switch to the battery voltage. As the V\textsubscript{BAT} voltage could be higher than V\textsubscript{DDA}, to ensure the correct operation of the ADC, the V\textsubscript{BAT} pin is internally connected to a bridge divider by 4. This bridge is automatically enabled when VBATEN is set, to connect V\textsubscript{BAT}/4 to the ADC input channels (see Table: ADC interconnection in Section 26.4.2: ADC pins and internal signals for more details). As a consequence, the converted digital value is one third of the V\textsubscript{BAT} voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the V\textsubscript{BAT}/4 voltage.

*Figure 257* shows the block diagram of the V\textsubscript{BAT} sensing feature.

\[
\text{Temperature (in °C)} = \frac{\text{TS\_CAL2\_TEMP} - \text{TS\_CAL1\_TEMP}}{\text{TS\_CAL2} - \text{TS\_CAL1}} \times (\text{TS\_DATA} - \text{TS\_CAL1}) + \text{TS\_CAL1\_TEMP}
\]
1. The VBATEN bit must be set to enable the conversion of internal channel for VBAT/4.

### 26.4.33 Monitoring the internal voltage reference

It is possible to monitor the internal voltage reference (VREFINT) to have a reference point for evaluating the ADC VREF+ voltage level.

Refer to Table: ADC interconnection in Section 26.4.2: ADC pins and internal signals for details on the ADC input channels to which the internal voltage reference is internally connected.

Refer to the electrical characteristics section of the product datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

*Figure 258* shows the block diagram of the VREFINT sensing feature.

1. The VREFEN bit into ADC_CCR register must be set to enable the conversion of internal channels (VREFINT).
Calculating the actual VREF+ voltage using the internal reference voltage

VREF+ voltage may be subject to variations or not precisely known. The embedded internal reference voltage VREFINT and its calibration data acquired by the ADC during the manufacturing process at VREF+_charac can be used to evaluate the actual VREF+ voltage level.

The following formula gives the actual VREF+ voltage supplying the device:

\[ V_{\text{REF+}} = V_{\text{REF+_Charac}} \times \frac{\text{VREFINT}_\text{CAL}}{\text{VREFINT}_\text{DATA}} \]

Where:

- \( V_{\text{REF+_Charac}} \) is the value of VREF+ voltage characterized at VREFINT during the manufacturing process. It is specified in the device datasheet.
- \( \text{VREFINT}_\text{CAL} \) is the VREFINT calibration value
- \( \text{VREFINT}_\text{DATA} \) is the actual VREFINT output value converted by ADC

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between VREF+ and the voltage applied on the converted channel.

For applications where VREF+ value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from VREF+:

\[ V_{\text{CHANNEL}_x} = \frac{V_{\text{REF+}} \times \text{ADC}_\text{DATA}}{\text{FULL}_\text{SCALE}} \]

By replacing VREF+ by the formula provided above, the absolute voltage value is given by the following formula

\[ V_{\text{CHANNEL}_x} = \frac{V_{\text{REF+_Charac}} \times \text{VREFINT}_\text{CAL} \times \text{ADC}_\text{DATA}}{\text{VREFINT}_\text{DATA} \times \text{FULL}_\text{SCALE}} \]

Where:

- \( V_{\text{REF+_Charac}} \) is the value of VREF+ voltage characterized at VREFINT during the manufacturing process.
- \( \text{VREFINT}_\text{CAL} \) is the VREFINT calibration value
- \( \text{ADC}_\text{DATA} \) is the value measured by the ADC on channel x (right-aligned)
- \( \text{VREFINT}_\text{DATA} \) is the actual VREFINT output value converted by the ADC
- \( \text{FULL}_\text{SCALE} \) is the maximum digital value of the ADC output. For example with 12-bit resolution, it is \( 2^{12} - 1 = 4095 \) or with 8-bit resolution, \( 2^8 - 1 = 255 \).

Note: If ADC measurements are done using an output format other than 12-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

26.4.34 Monitoring the supply voltage

ADC2 is connected to the internal supply voltage. To use the ADC to measure this voltage, enable the connection through ADC option register.
26.5 ADC interrupts

For each ADC, an interrupt can be generated:
- After ADC power-up, when the ADC is ready (flag ADRDY)
- On the end of any conversion for regular groups (flag EOC)
- On the end of a sequence of conversion for regular groups (flag EOS)
- On the end of any conversion for injected groups (flag JEOC)
- On the end of a sequence of conversion for injected groups (flag JEOS)
- When an analog watchdog detection occurs (flag AWD1, AWD2 and AWD3)
- When the end of sampling phase occurs (flag EOSMP)
- When the data overrun occurs (flag OVR)
- When the injected sequence context queue overflows (flag JQOVF)

Separate interrupt enable bits are available for flexibility.

Table 252. ADC interrupts

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop, Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>ADC ready</td>
<td>ADRDY</td>
<td>ADRDYIE</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>End of conversion of a regular group</td>
<td>EOC</td>
<td>EOCIE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>End of conversion sequence of a regular group</td>
<td>EOS</td>
<td>EOSIE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>End of conversion of an injected group</td>
<td>JEOC</td>
<td>JEOCIE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>End of conversion sequence of an injected group</td>
<td>JEOS</td>
<td>JEOSIE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Analog watchdog 1 status bit is set</td>
<td>AWD1</td>
<td>AWD1IE</td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Analog watchdog 2 status bit is set</td>
<td>AWD2</td>
<td>AWD2IE</td>
<td></td>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Analog watchdog 3 status bit is set</td>
<td>AWD3</td>
<td>AWD3IE</td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>End of sampling phase</td>
<td>EOSMP</td>
<td>EOSMPIE</td>
<td></td>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Overrun</td>
<td>OVR</td>
<td>OVRIE</td>
<td></td>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Injected context queue overflows</td>
<td>JQOVF</td>
<td>JQOVFIE</td>
<td></td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>

Set by hardware and cleared by software.
### 26.6 ADC registers (for each ADC)

Refer to Section 1.2 on page 101 for a list of abbreviations used in register descriptions.

#### 26.6.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>JQOVF</th>
<th>AWD3</th>
<th>AWD2</th>
<th>AWD1</th>
<th>JEOS</th>
<th>JEOC</th>
<th>OVR</th>
<th>EOS</th>
<th>EOC</th>
<th>EOSMP</th>
<th>ADRDY</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.

- **Bit 10 JQOVF: Injected context queue overflow**
  - This bit is set by hardware when an Overflow of the Injected Queue of Context occurs. It is cleared by software writing 1 to it. Refer to Section 26.4.21: Queue of context for injected conversions for more information.
  - 0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)
  - 1: Injected context queue overflow has occurred

- **Bit 9 AWD3: Analog watchdog 3 flag**
  - This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADC_TR3 register. It is cleared by software writing 1 to it.
  - 0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)
  - 1: Analog watchdog 3 event occurred

- **Bit 8 AWD2: Analog watchdog 2 flag**
  - This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADC_TR2 register. It is cleared by software writing 1 to it.
  - 0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)
  - 1: Analog watchdog 2 event occurred

- **Bit 7 AWD1: Analog watchdog 1 flag**
  - This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADC_TR1 register. It is cleared by software, writing 1 to it.
  - 0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)
  - 1: Analog watchdog 1 event occurred

- **Bit 6 JEOS: Injected channel end of sequence flag**
  - This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.
  - 0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)
  - 1: Injected conversions complete
Bit 5  **JEOC**: Injected channel end of conversion flag
- This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADC_JDRy register.
- 0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)
- 1: Injected channel conversion complete

Bit 4  **OVR**: ADC overrun
- This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed while the EOC flag was already set. It is cleared by software writing 1 to it.
- 0: No overrun occurred (or the flag event was already acknowledged and cleared by software)
- 1: Overrun has occurred

Bit 3  **EOS**: End of regular sequence flag
- This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.
- 0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)
- 1: Regular Conversions sequence complete

Bit 2  **EOC**: End of conversion flag
- This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register.
- 0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)
- 1: Regular channel conversion complete

Bit 1  **EOSMP**: End of sampling flag
- This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.
- 0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)
- 1: End of sampling phase reached

Bit 0  **ADRDY**: ADC ready
- This bit is set by hardware after the ADC has been enabled (ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests.
- It is cleared by software writing 1 to it.
- 0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)
- 1: ADC is ready to start conversion
## 26.6.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x04  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:11</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>JQOVFIE: Injected context queue overflow interrupt enable</td>
<td>rw</td>
<td>The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).</td>
</tr>
<tr>
<td>9</td>
<td>AWD3IE: Analog watchdog 3 interrupt enable</td>
<td>rw</td>
<td>The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).</td>
</tr>
<tr>
<td>8</td>
<td>AWD2IE: Analog watchdog 2 interrupt enable</td>
<td>rw</td>
<td>The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).</td>
</tr>
<tr>
<td>7</td>
<td>AWD1IE: Analog watchdog 1 interrupt enable</td>
<td>rw</td>
<td>The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).</td>
</tr>
<tr>
<td>6</td>
<td>JEOSIE: End of injected sequence of conversions interrupt enable</td>
<td>rw</td>
<td>The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).</td>
</tr>
</tbody>
</table>
Bit 5  **JEOCIE**: End of injected conversion interrupt enable
This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.
0: JEOC interrupt disabled.
1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.
*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 4  **OVRIE**: Overrun interrupt enable
This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.
0: Overrun interrupt disabled
1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.
*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 3  **EOSIE**: End of regular sequence of conversions interrupt enable
This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.
0: EOS interrupt disabled
1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.
*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 2  **EOCIE**: End of regular conversion interrupt enable
This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.
0: EOC interrupt disabled.
1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.
*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 1  **EOSMPIE**: End of sampling flag interrupt enable for regular conversions
This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.
0: EOSMP interrupt disabled.
1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.
*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 0  **ADRDYIE**: ADC ready interrupt enable
This bit is set and cleared by software to enable/disable the ADC Ready interrupt.
0: ADRDY interrupt disabled
1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.
*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*
### 26.6.3 ADC control register (ADC_CR)

Address offset: 0x08  
Reset value: 0x2000 0000

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

#### Bit 31 ADCAL: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or Differential inputs mode.

- It is cleared by hardware after calibration is complete.
- 0: Calibration complete
- 1: Write 1 to calibrate the ADC. Read at 1 means that a calibration in progress.

**Note:** The software is allowed to launch a calibration by setting ADCAL only when ADEN = 0.

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).

#### Bit 30 ADCALDIF: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or Differential inputs mode for the calibration.

- 0: Writing ADCAL launches a calibration in single-ended inputs mode.
- 1: Writing ADCAL launches a calibration in Differential inputs mode.

**Note:** The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

#### Bit 29 DEEPPWD: Deep-power-down enable

This bit is set and cleared by software to put the ADC in Deep-power-down mode.

- 0: ADC not in Deep-power down
- 1: ADC in Deep-power-down (default reset state)

**Note:** The software is allowed to write this bit only when the ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).
Bit 28 **ADVREGEN**: ADC voltage regulator enable

This bit is set by software to enable the ADC voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC Voltage regulator disabled
1: ADC Voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to Section 26.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN).

The software can program this bit field only when the ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be re-configured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

0: No ADC stop injected conversion command ongoing
1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

**Note:** The software is allowed to set JADSTP only when JADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting an injected conversion and there is no pending request to disable the ADC)

In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

0: No ADC stop regular conversion command ongoing
1: Write 1 to stop regular conversions ongoing. Read 1 means that an ADSTP command is in progress.

**Note:** The software is allowed to set ADSTP only when ADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting a regular conversion and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).
Bit 3  **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN, a conversion immediately starts (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- in single conversion mode when software trigger is selected (JEXTSEL = 0x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.
- in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.
1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

**Note:** The software is allowed to set JADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)
Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN, a conversion immediately starts (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- in single conversion mode when software trigger is selected (EXTSEL = 0x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.
- in all cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.
1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

**Note:** The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC)

In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing
1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

**Note:** The software is allowed to set ADDIS only when ADEN = 1 and both ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the flag ADRDY has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)
1: Write 1 to enable the ADC.

**Note:** The software is allowed to set ADEN only when all bits of ADC_CR registers are 0 (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0) except for bit ADVREGEN which must be 1 (and the software must have wait for the startup time of the voltage regulator)
26.6.4 ADC configuration register (ADC CfGR)

Address offset: 0x0C
Reset value: 0x8000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>JQDIS</td>
<td>AWD1CH[4:0]</td>
<td>JAUTO</td>
<td>JAWD1EN</td>
<td>AWDO1EN</td>
<td>AWD1SQL</td>
<td>JOM</td>
<td>JDISCEN</td>
<td>DISCNUM[2:0]</td>
<td>DISCEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31 JQDIS: Injected queue disable
This bit is set and cleared by software to disable the injected queue mechanism:
0: Injected queue enabled
1: Injected queue disabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing).
A set or reset of JQDIS bit causes the injected queue to be flushed and the JSQR register is cleared.

Bits 30:26 AWD1CH[4:0]: Analog watchdog 1 channel selection
These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.
00000: ADC analog input channel 0 monitored by AWD1 (available on ADC1 only)
00001: ADC analog input channel 1 monitored by AWD1
.....
10011: ADC analog input channel 19 monitored by AWD1
others: reserved, must not be used

Note: Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value.
The channel selected by AWD1CH must be also selected into the SO[i] or JSQRi registers.
The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 JAUTO: Automatic injected group conversion
This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.
0: Automatic injected group conversion disabled
1: Automatic injected group conversion enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing).
When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit JAUTO of the slave ADC is no more writable and its content is equal to the bit JAUTO of the master ADC.
Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels
This bit is set and cleared by software
0: Analog watchdog 1 disabled on injected channels
1: Analog watchdog 1 enabled on injected channels
*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels
This bit is set and cleared by software
0: Analog watchdog 1 disabled on regular channels
1: Analog watchdog 1 enabled on regular channels
*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels
This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels
0: Analog watchdog 1 enabled on all channels
1: Analog watchdog 1 enabled on a single channel
*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 21 **JQM**: JSQR queue mode
This bit is set and cleared by software.
It defines how an empty Queue is managed.
0: JSQR mode 0: The Queue is never empty and maintains the last written configuration into JSQR.
1: JSQR mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.
Refer to Section 26.4.21: Queue of context for injected conversions for more information.
*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit JQM of the slave ADC is no more writable and its content is equal to the bit JQM of the master ADC.

Bit 20 **JDISCEN**: Discontinuous mode on injected channels
This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.
0: Discontinuous mode on injected channels disabled
1: Discontinuous mode on injected channels enabled
*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.
When dual mode is enabled (bits DUAL of ADC_CCR register are not equal to zero), the bit JDISCEN of the slave ADC is no more writable and its content is equal to the bit JDISCEN of the master ADC.
Bits 19:17 **DISCNUM[2:0]**: Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel
001: 2 channels
...
111: 8 channels

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bits DISCNUM[2:0] of the slave ADC are no more writable and their content is equal to the bits DISCNUM[2:0] of the master ADC.

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled
1: Discontinuous mode for regular channels enabled

*Note:* It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit DISCEN of the slave ADC is no more writable and its content is equal to the bit DISCEN of the master ADC.

Bit 15 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to Section: Data register, data alignment and offset (ADC_DR, OFFSET, OFFSET_CH, ALIGN).

0: Right alignment
1: Left alignment

*Note:* The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the Auto Delayed Conversion mode.

0: Auto-delayed conversion mode off
1: Auto-delayed conversion mode on

*Note:* The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit AUTDLY of the slave ADC is no more writable and its content is equal to the bit AUTDLY of the master ADC.
Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

0: Single conversion mode
1: Continuous conversion mode

**Note:** *It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.*

The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit CONT of the slave ADC is no more writable and its content is equal to the bit CONT of the master ADC.

Bit 12 **OVRMOD**: Overrun mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.
1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

**Note:** The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)
01: Hardware trigger detection on the rising edge
10: Hardware trigger detection on the falling edge
11: Hardware trigger detection on both the rising and falling edges

**Note:** The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

00000: adc_ext_trg0
00001: adc_ext_trg1
00010: adc_ext_trg2
00011: adc_ext_trg3
00100: adc_ext_trg4
00101: adc_ext_trg5
00110: adc_ext_trg6
00111: adc_ext_trg7
...
11111: adc_ext_trg31

**Note:** The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 4:3 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit
01: 10-bit
10: 8-bit
11: 6-bit

**Note:** The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
### 26.6.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10

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

#### Bit 2
Reserved, must be kept at reset value.

#### Bit 1 DMACFG: Direct memory access configuration
This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN = 1.
- 0: DMA One Shot mode selected
- 1: DMA Circular mode selected

For more details, refer to Section : Managing conversions using the DMA

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

In dual-ADC modes, this bit is not relevant and replaced by control bit DMACFG of the ADC_CCR register.

#### Bit 0 DMAEN: Direct memory access enable
This bit is set and cleared by software to enable the generation of DMA requests. This allows to use the DMA to manage automatically the converted data. For more details, refer to Section : Managing conversions using the DMA.
- 0: DMA disabled
- 1: DMA enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

In dual-ADC modes, this bit is not relevant and replaced by control bits MDMA[1:0] of the ADC_CCR register.
Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **SMPTRIG**: Sampling time control trigger mode

This bit is set and cleared by software to enable the sampling time control trigger mode.

- 0: Sampling time control trigger mode disabled
- 1: Sampling time control trigger mode enabled

The sampling time starts on the trigger rising edge, and the conversion on the trigger falling edge.

EXTEN bit should be set to 01. BULB bit must not be set when the SMPTRIG bit is set.

When EXTEN bit is set to 00, set SWTRIG to start the sampling and clear SWTRIG bit to start the conversion.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 26 **BULB**: Bulb sampling mode

This bit is set and cleared by software to enable the bulb sampling mode.

- 0: Bulb sampling mode disabled
- 1: Bulb sampling mode enabled. The sampling period starts just after the previous end of conversion.

SAMPTRIG bit must not be set when the BULB bit is set.

The very first ADC conversion is performed with the sampling time specified in SMPx bits.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 25 **SWTRIG**: Software trigger bit for sampling time control trigger mode

This bit is set and cleared by software to enable the bulb sampling mode.

- 0: Software trigger starts the conversion for sampling time control trigger mode
- 1: Software trigger starts the sampling for sampling time control trigger mode

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 24:17 Reserved, must be kept at reset value.

Bits 16:11 Reserved, must be kept at reset value.

Bit 10 **ROVSM**: Regular oversampling mode

This bit is set and cleared by software to select the regular oversampling mode.

- 0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)
- 1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 9 **TROVS**: Triggered Regular oversampling

This bit is set and cleared by software to enable triggered oversampling.

- 0: All oversampled conversions for a channel are done consecutively following a trigger
- 1: Each oversampled conversion for a channel needs a new trigger

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*
Bits 8:5 **OVSS[3:0]:** Oversampling shift  
This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.  
0000: No shift  
0001: Shift 1-bit  
0010: Shift 2-bits  
0011: Shift 3-bits  
0100: Shift 4-bits  
0101: Shift 5-bits  
0110: Shift 6-bits  
0111: Shift 7-bits  
1000: Shift 8-bits  
Other codes reserved  
*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 4:2 **OVS[2:0]:** Oversampling ratio  
This bitfield is set and cleared by software to define the oversampling ratio.  
000: 2x  
001: 4x  
010: 8x  
011: 16x  
100: 32x  
101: 64x  
110: 128x  
111: 256x  
*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 1 **JOVSE:** Injected oversampling Enable  
This bit is set and cleared by software to enable injected oversampling.  
0: Injected oversampling disabled  
1: Injected oversampling enabled  
*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)*

Bit 0 **ROVSE:** Regular oversampling Enable  
This bit is set and cleared by software to enable regular oversampling.  
0: Regular oversampling disabled  
1: Regular oversampling enabled  
*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)*
### 26.6.6 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x14

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
</tr>
</tbody>
</table>

Bit 31 **SMPPLUS**: Addition of one clock cycle to the sampling time.
- 1: 2.5 ADC clock cycle sampling time becomes 3.5 ADC clock cycles for the ADC_SMPR1 and ADC_SMPR2 registers.
- 0: The sampling time remains set to 2.5 ADC clock cycles remains

To make sure no conversion is ongoing, the software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0.

Bit 30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]**: Channel x sampling time selection (x = 9 to 0)

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

- 000: 2.5 ADC clock cycles
- 001: 6.5 ADC clock cycles
- 010: 12.5 ADC clock cycles
- 011: 24.5 ADC clock cycles
- 100: 47.5 ADC clock cycles
- 101: 92.5 ADC clock cycles
- 110: 247.5 ADC clock cycles
- 111: 640.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

### 26.6.7 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x18

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
</tr>
</tbody>
</table>

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
26.6.8 ADC watchdog threshold register 1 (ADC_TR1)

Address offset: 0x20

Reset value: 0x0FFF 0000

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


Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]**: Channel x sampling time selection (x = 19 to 10)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

- 000: 2.5 ADC clock cycles
- 001: 6.5 ADC clock cycles
- 010: 12.5 ADC clock cycles
- 011: 24.5 ADC clock cycles
- 100: 47.5 ADC clock cycles
- 101: 92.5 ADC clock cycles
- 110: 247.5 ADC clock cycles
- 111: 640.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing). Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 1. Refer to Section 26.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
Bit 15  Reserved, must be kept at reset value.

Bits 14:12  **AWDFILT[2:0]**: Analog watchdog filtering parameter
   This bit is set and cleared by software.
   000: No filtering
   001: two consecutive detection generates an AWDx flag or an interrupt
   ...
   111: Eight consecutive detection generates an AWDx flag or an interrupt

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 11:0  **LT1[11:0]**: Analog watchdog 1 lower threshold
   These bits are written by software to define the lower threshold for the analog watchdog 1.
   Refer to Section 26.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

### 26.6.9 ADC watchdog threshold register 2 (ADC_TR2)

Address offset: 0x24

Reset value: 0x00FF 0000

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

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

Bits 31:24  Reserved, must be kept at reset value.

Bits 23:16  **HT2[7:0]**: Analog watchdog 2 higher threshold
   These bits are written by software to define the higher threshold for the analog watchdog 2.
   Refer to Section 26.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 15:8  Reserved, must be kept at reset value.

Bits 7:0  **LT2[7:0]**: Analog watchdog 2 lower threshold
   These bits are written by software to define the lower threshold for the analog watchdog 2.
   Refer to Section 26.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*
### 26.6.10 ADC watchdog threshold register 3 (ADC_TR3)

Address offset: 0x28
Reset value: 0x00FF 0000

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HT3[7:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 3. Refer to Section 26.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **LT3[7:0]**: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 3. This watchdog compares the 8-bit of LT3 with the 8 MSB of the converted data.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

### 26.6.11 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x30
Reset value: 0x0000 0000

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]**: 4th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 4th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.
Bits 22:18 **SQ3[4:0]**: 3rd conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 3rd in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]**: 2nd conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 2nd in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]**: 1st conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 1st in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **L[3:0]**: Regular channel sequence length
These bits are written by software to define the total number of conversions in the regular channel conversion sequence.

- 0000: 1 conversion
- 0001: 2 conversions
- ...
- 1111: 16 conversions

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

*Note:* Some channels are not connected physically and must not be selected for conversion.

### 26.6.12 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x34

Reset value: 0x0000 0000
Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]**: 9th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 9th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]**: 8th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 8th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]**: 7th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 7th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]**: 6th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 6th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]**: 5th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 5th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

*Note:* Some channels are not connected physically and must not be selected for conversion.

### 26.6.13 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x38

Reset value: 0x0000 0000
Bits 31:29  Reserved, must be kept at reset value.

Bits 28:24  **SQ14[4:0]**: 14th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 14th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23  Reserved, must be kept at reset value.

Bits 22:18  **SQ13[4:0]**: 13th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 13th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17  Reserved, must be kept at reset value.

Bits 16:12  **SQ12[4:0]**: 12th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 12th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11  Reserved, must be kept at reset value.

Bits 10:6  **SQ11[4:0]**: 11th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 11th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5  Reserved, must be kept at reset value.

Bits 4:0  **SQ10[4:0]**: 10th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 10th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

*Note:* Some channels are not connected physically and must not be selected for conversion.

### 26.6.14 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x3C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:11  Reserved, must be kept at reset value.

Bits 10:6  SQ16[4:0]: 16th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 16th in
the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures
that no regular conversion is ongoing).

Bit 5   Reserved, must be kept at reset value.

Bits 4:0  SQ15[4:0]: 15th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 15th in
the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures
that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

26.6.15  ADC regular data register (ADC_DR)
Address offset: 0x40
Reset value: 0x0000 0000

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

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

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  RDATA[15:0]:  Regular data converted
These bits are read-only. They contain the conversion result from the last converted regular channel.
The data are left- or right-aligned as described in Section 26.4.26: Data management.

26.6.16  ADC injected sequence register (ADC_JSQR)
Address offset: 0x4C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Bits 31:27 \textbf{JSQ4}[4:0]: 4th conversion in the injected sequence
These bits are written by software with the channel number (0 to 19) assigned as the 4th in
the injected conversion sequence.
\textit{Note:} The software is allowed to write these bits only when \textit{JADSTART} = 0 (which ensures
that no injected conversion is ongoing).

Bit 26 Reserved, must be kept at reset value.

Bits 25:21 \textbf{JSQ3}[4:0]: 3rd conversion in the injected sequence
These bits are written by software with the channel number (0 to 19) assigned as the 3rd in
the injected conversion sequence.
\textit{Note:} The software is allowed to write these bits only when \textit{JADSTART} = 0 (which ensures
that no injected conversion is ongoing).

Bit 20 Reserved, must be kept at reset value.

Bits 19:15 \textbf{JSQ2}[4:0]: 2nd conversion in the injected sequence
These bits are written by software with the channel number (0 to 19) assigned as the 2nd in
the injected conversion sequence.
\textit{Note:} The software is allowed to write these bits only when \textit{JADSTART} = 0 (which ensures
that no injected conversion is ongoing).

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 \textbf{JSQ1}[4:0]: 1st conversion in the injected sequence
These bits are written by software with the channel number (0 to 19) assigned as the 1st in
the injected conversion sequence.
\textit{Note:} The software is allowed to write these bits only when \textit{JADSTART} = 0 (which ensures
that no injected conversion is ongoing).
Bits 8:7 **JEXTEN[1:0]**: External trigger enable and polarity selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: If JQDIS = 0 (queue enabled), hardware and software trigger detection disabled. Otherwise, the queue is disabled as well as hardware trigger detection (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

*Note:* The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

If JQM = 1 and if the Queue of Context becomes empty, the software and hardware triggers of the injected sequence are both internally disabled (refer to Section 26.4.21: Queue of context for injected conversions)

Bits 6:2 **JEXTSEL[4:0]**: External Trigger Selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: adc_jext_trg0

00001: adc_jext_trg1

00010: adc_jext_trg2

00011: adc_jext_trg3

00100: adc_jext_trg4

00101: adc_jext_trg5

00110: adc_jext_trg6

00111: adc_jext_trg7

... 11111: adc_jext_trg31

*Note:* The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 1:0 **JL[1:0]**: Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

*Note:* The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

*Note:* Some channels are not connected physically and must not be selected for conversion.
26.6.17 ADC offset y register (ADC_OFRy)

Address offset: $0x60 + 0x04 \times (y - 1)$, ($y = 1$ to $4$)

Reset value: $0x0000 0000$

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31  | OFFSET_EN: Offset y enable   | This bit is written by software to enable or disable the offset programmed into bits OFFSET[11:0].
|     |                              | *Note:* The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
| 30-26 | OFFSET_CH[4:0]: Channel selection for the data offset y | These bits are written by software to define the channel to which the offset programmed into bits OFFSET[11:0] applies.
|     |                              | *Note:* The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
|     |                              | Some channels are not connected physically and must not be selected for the data offset y.
|     |                              | If OFFSET_EN is set, it is not allowed to select the same channel for different ADC_OFRy registers.
| 25  | SATEN: Saturation enable    | This bit is set and cleared by software to enable the saturation at 0x000 and 0xFFF for the offset function.
|     |                              | 0: No saturation control, offset result can be signed
|     |                              | 1: Saturation enabled, offset result unsigned and saturated at 0x000 and 0xFFF
|     |                              | *Note:* The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
Bit 24 **OFFSETPOS**: Positive offset
This bit is set and cleared by software to enable the positive offset.
0: Negative offset
1: Positive offset

*Note:* The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:0 **OFFSET[11:0]**: Data offset y for the channel programmed into bits OFFSET_CH[4:0]
These bits are written by software to define the offset to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset must be programmed in the bits OFFSET_CH[4:0]. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

*Note:* The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

If several offset (OFFSET) point to the same channel, only the offset with the lowest x value is considered for the subtraction.

*Ex:* if OFFSET1_CH[4:0] = 4 and OFFSET2_CH[4:0] = 4, this is OFFSET1[11:0] which is subtracted when converting channel 4.

### 26.6.18 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x80 + 0x04 * (y - 1), (y = 1 to 4)
Reset value: 0x0000 0000

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

**JDATA[15:0]**

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **JDATA[15:0]**: Injected data
These bits are read-only. They contain the conversion result from injected channel y. The data are left -or right-aligned as described in *Section 26.4.26: Data management.*
### 26.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR)

Address offset: 0xA0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWD2CH[19:16]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>AWD2CH[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWD2CH[19:0]:** Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

- **AWD2CH[i] = 0:** ADC analog input channel i is not monitored by AWD2
- **AWD2CH[i] = 1:** ADC analog input channel i is monitored by AWD2

When **AWD2CH[19:0] = 000..0**, the analog Watchdog 2 is disabled

**Note:** The channels selected by **AWD2CH** must be also selected into the SQRi or JSQRi registers.

The software is allowed to write these bits only when **ADSTART = 0** and **JADSTART = 0** (which ensures that no conversion is ongoing).

Some channels are not connected physically and must not be selected for the analog watchdog.

### 26.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR)

Address offset: 0xA4  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWD3CH[19:16]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>AWD3CH[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

|            | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  |

|            | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  |

|            | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  |

|            | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  |
26.6.21 ADC Differential mode selection register (ADC_DIFSEL)

Address offset: 0xB0

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 DIFSEL[19:0]: Differential mode for channels 19 to 0.

These bits are set and cleared by software. They allow to select if a channel is configured as single-ended or Differential mode.

DIFSEL[i] = 0: ADC analog input channel is configured in single-ended mode
DIFSEL[i] = 1: ADC analog input channel i is configured in Differential mode

Note: The DIFSEL bits corresponding to channels that are either connected to a single-ended I/O port or to an internal channel must be kept their reset value (single-ended input mode).

The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).
### 26.6.22 ADC calibration factors (ADC_CALFACT)

Address offset: 0xB4
Reset value: 0x0000 0000

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

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

Bits 22:16 **CALFACT_D[6:0]**: Calibration Factors in differential mode
- These bits are written by hardware or by software.
- Once a differential inputs calibration is complete, they are updated by hardware with the calibration factors.
- Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new differential calibration is launched.
- **Note:** The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

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

Bits 6:0 **CALFACT_S[6:0]**: Calibration Factors in single-ended mode
- These bits are written by hardware or by software.
- Once a single-ended inputs calibration is complete, they are updated by hardware with the calibration factors.
- Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new single-ended calibration is launched.
- **Note:** The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

### 26.6.23 ADC option register (ADC_OR)

Address offset: 0xC8
Reset value: 0x0000 0000

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

Bits 15:7 Reserved, must be kept at reset value.
26.7 ADC common registers

These registers define the control and status registers common to master and slave ADCs:

### 26.7.1 ADC common status register (ADC_CSR)

Address offset: 0x300
Reset value: 0x0000 0000

This register provides an image of the status bits of the different ADC. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing 0 to it in the corresponding ADC_ISR register.

| Bit 31  | Bit 30  | Bit 29  | Bit 28  | Bit 27  | Bit 26  | Bit 25  | Bit 24  | Bit 23  | Bit 22  | Bit 21  | Bit 20  | Bit 19  | Bit 18  | Bit 17  | Bit 16  |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| Reserved | Reserved | Reserved | Reserved | Reserved | JQOVF_SLV | AWD3_SLV | AWD2_SLV | AWD1_SLV | JEOS_SLV | JEOC_SLV | OVR_SLV | EOS_SLV | EOC_SLV | EOSMP_SLV | ADRDY_SLV |
| r      | r      | r      | r      | r      | r      | r      | r      | r      | r      | r      | r      | r      | r      | r      |
| 15     | 14     | 13     | 12     | 11     | 10     | 9      | 8      | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
| Reserved | Reserved | Reserved | Reserved | Reserved | JQOVF_MST | AWD3_MST | AWD2_MST | AWD1_MST | JEOS_MST | JEOC_MST | OVR_MST | EOS_MST | EOC_MST | EOSMP_MST | ADRDY_MST |
| r      | r      | r      | r      | r      | r      | r      | r      | r      | r      | r      | r      | r      | r      | r      |

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

- **Bit 26 JQOVF_SLV**: Injected Context Queue Overflow flag of the slave ADC
  - This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

- **Bit 25 AWD3_SLV**: Analog watchdog 3 flag of the slave ADC
  - This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

- **Bit 24 AWD2_SLV**: Analog watchdog 2 flag of the slave ADC
  - This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

- **Bit 23 AWD1_SLV**: Analog watchdog 1 flag of the slave ADC
  - This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

- **Bit 22 JEOS_SLV**: End of injected sequence flag of the slave ADC
  - This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.
Bit 21  **JEOC_SLV**: End of injected conversion flag of the slave ADC  
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 20  **OVR_SLV**: Overrun flag of the slave ADC  
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 19  **EOS_SLV**: End of regular sequence flag of the slave ADC. This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 18  **EOC_SLV**: End of regular conversion of the slave ADC  
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 17  **EOSMP_SLV**: End of Sampling phase flag of the slave ADC  
This bit is a copy of the EOSMP2 bit in the corresponding ADC_ISR register.

Bit 16  **ADRDY_SLV**: Slave ADC ready  
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

Bits 15:11  Reserved, must be kept at reset value.

Bit 10  **JQOVF_MST**: Injected Context Queue Overflow flag of the master ADC  
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 9  **AWD3_MST**: Analog watchdog 3 flag of the master ADC  
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8  **AWD2_MST**: Analog watchdog 2 flag of the master ADC  
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7  **AWD1_MST**: Analog watchdog 1 flag of the master ADC  
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6  **JEOS_MST**: End of injected sequence flag of the master ADC  
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5  **JEOC_MST**: End of injected conversion flag of the master ADC  
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4  **OVR_MST**: Overrun flag of the master ADC  
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3  **EOS_MST**: End of regular sequence flag of the master ADC  
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2  **EOC_MST**: End of regular conversion of the master ADC  
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1  **EOSMP_MST**: End of Sampling phase flag of the master ADC  
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0  **ADRDY_MST**: Master ADC ready  
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.
### 26.7.2 ADC common control register (ADC_CCR)

Address offset: 0x308  
Reset value: 0x0000 0000

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


Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATEN**: VBAT enable  
This bit is set and cleared by software to control.  
0: VBAT channel disabled  
1: VBAT channel enabled

Bit 23 **TSEN**: VSENSE enable  
This bit is set and cleared by software to control VSENSE.  
0: Temperature sensor channel disabled  
1: Temperature sensor channel enabled

Bit 22 **VREFEN**: VREFINT enable  
This bit is set and cleared by software to enable/disable the VREFINT channel.  
0: VREFINT channel disabled  
1: VREFINT channel enabled

Bits 21:18 **PRESC[3:0]**: ADC prescaler  
These bits are set and cleared by software to select the frequency of the clock to the ADC.  
The clock is common for all the ADCs.  
0000: input ADC clock not divided  
0001: input ADC clock divided by 2  
0010: input ADC clock divided by 4  
0011: input ADC clock divided by 6  
0100: input ADC clock divided by 8  
0101: input ADC clock divided by 10  
0110: input ADC clock divided by 12  
0111: input ADC clock divided by 16  
1000: input ADC clock divided by 32  
1001: input ADC clock divided by 64  
1010: input ADC clock divided by 128  
1011: input ADC clock divided by 256  
other: reserved

*Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0). The ADC prescaler value is applied only when CKMODE[1:0] = 0b00.*
Bits 17:16 **CKMODE[1:0]: ADC clock mode**

These bits are set and cleared by software to define the ADC clock scheme (which is common to both master and slave ADCs):

- 00: adc_ker_ck (x = 1/2) (Asynchronous clock mode), generated at product level (refer to Section 6: Reset and clock control (RCC))
- 01: adc_hclk/1 (Synchronous clock mode). This configuration must be enabled only if the AHB clock prescaler is set to 1 (HPRE[3:0] = 0XXX in RCC_CFGR register) and if the system clock has a 50% duty cycle.
- 10: adc_hclk/2 (Synchronous clock mode)
- 11: adc_hclk/4 (Synchronous clock mode)

In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.

*Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).*

Bits 15:14 **MDMA[1:0]: Direct memory access mode for dual ADC mode**

This bitfield is set and cleared by software. Refer to the DMA controller section for more details.

- 00: MDMA mode disabled
- 01: Reserved
- 10: MDMA mode enabled for 12 and 10-bit resolution
- 11: MDMA mode enabled for 8 and 6-bit resolution

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 13 **DMACFG: DMA configuration (for dual ADC mode)**

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN = 1.

- 0: DMA One Shot mode selected
- 1: DMA Circular mode selected

For more details, refer to Section : Managing conversions using the DMA

*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 12 **Reserved, must be kept at reset value.**
Bits 11:8 **DELAY[3:0]**: Delay between 2 sampling phases

These bits are set and cleared by software. These bits are used in dual interleaved modes. Refer to Table 253 for the value of ADC resolution versus DELAY bits values.

*Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).*

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DUAL[4:0]**: Dual ADC mode selection

These bits are written by software to select the operating mode. 00000 corresponds to independent mode. Values 00001 to 01001 correspond to dual mode, master and slave ADCs working together.

- 00000: Independent mode
- 00001: Combined regular simultaneous + injected simultaneous mode
- 00010: Combined regular simultaneous + alternate trigger mode
- 00011: Combined interleaved mode + injected simultaneous mode
- 00100: Reserved
- 00101: Injected simultaneous mode only
- 00110: Regular simultaneous mode only
- 00111: Interleaved mode only
- 01001: Alternate trigger mode only
- Others: Reserved, must not be used

*Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).*

<table>
<thead>
<tr>
<th>DELAY bits</th>
<th>12-bit resolution</th>
<th>10-bit resolution</th>
<th>8-bit resolution</th>
<th>6-bit resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1 * Tadc_ker_ck</td>
<td>1 * Tadc_ker_ck</td>
<td>1 * Tadc_ker_ck</td>
<td>1 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0001</td>
<td>2 * Tadc_ker_ck</td>
<td>2 * Tadc_ker_ck</td>
<td>2 * Tadc_ker_ck</td>
<td>2 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0010</td>
<td>3 * Tadc_ker_ck</td>
<td>3 * Tadc_ker_ck</td>
<td>3 * Tadc_ker_ck</td>
<td>3 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0011</td>
<td>4 * Tadc_ker_ck</td>
<td>4 * Tadc_ker_ck</td>
<td>4 * Tadc_ker_ck</td>
<td>4 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0100</td>
<td>5 * Tadc_ker_ck</td>
<td>5 * Tadc_ker_ck</td>
<td>5 * Tadc_ker_ck</td>
<td>5 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0101</td>
<td>6 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0110</td>
<td>7 * Tadc_ker_ck</td>
<td>7 * Tadc_ker_ck</td>
<td>7 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0111</td>
<td>8 * Tadc_ker_ck</td>
<td>8 * Tadc_ker_ck</td>
<td>8 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
<tr>
<td>1000</td>
<td>9 * Tadc_ker_ck</td>
<td>9 * Tadc_ker_ck</td>
<td>8 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
<tr>
<td>1001</td>
<td>10 * Tadc_ker_ck</td>
<td>10 * Tadc_ker_ck</td>
<td>8 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
<tr>
<td>1010</td>
<td>11 * Tadc_ker_ck</td>
<td>10 * Tadc_ker_ck</td>
<td>8 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
<tr>
<td>1011</td>
<td>12 * Tadc_ker_ck</td>
<td>10 * Tadc_ker_ck</td>
<td>8 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
<tr>
<td>others</td>
<td>12 * Tadc_ker_ck</td>
<td>10 * Tadc_ker_ck</td>
<td>8 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
</tbody>
</table>
26.7.3 ADC common regular data register for dual mode (ADC_CDR)

Address offset: 0x30C
Reset value: 0x0000 0000

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

Bits 31:16 **RDATA_SLV[15:0]**: Regular data of the slave ADC
In dual mode, these bits contain the regular data of the slave ADC. Refer to Section 26.4.30: Dual ADC modes.
The data alignment is applied as described in Section : Data register, data alignment and offset (ADC_DR, OFFSET, OFFSET_CH, ALIGN)

Bits 15:0 **RDATA_MST[15:0]**: Regular data of the master ADC.
In dual mode, these bits contain the regular data of the master ADC. Refer to Section 26.4.30: Dual ADC modes.
The data alignment is applied as described in Section : Data register, data alignment and offset (ADC_DR, OFFSET, OFFSET_CH, ALIGN)
In MDMA = 0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].

26.7.4 ADC hardware configuration register (ADC_HWCFCGR0)

Address offset: 0x3F0
Reset value: 0x0000 1212

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IDLEVALUE[3:0]**: Idle value for non-selected channels
0000: Dummy channel selection is 0x13
0001: Dummy channel selection is 0x1F
26.7.5 ADC version register (ADC_VERR)

Address offset: 0x3F4
Reset value: 0x0000 0012

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

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

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

Bits 7:4 **MAJREV[3:0]**: Major revision
These bits returns the ADC IP major revision
0001: Major revision = 1.X

Bits 3:0 **MINREV[3:0]**: Minor revision
These bits returns the ADC IP minor revision
0001: Minor revision = X.1
0002: Minor revision = X.2
0003: Minor revision = X.3

26.7.6 ADC identification register (ADC_IPDR)

Address offset: 0x3F8
Reset value: 0x0011 0006

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID[15:0]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
26.7.7 ADC size identification register (ADC_SIDR)

Address offset: 0x3FC
Reset value: 0xA3C5 DD01

<table>
<thead>
<tr>
<th></th>
<th>SID[31:16]</th>
<th>SID[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>30</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>29</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>28</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>27</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>26</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>25</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>24</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>23</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>22</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>21</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>20</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>18</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>17</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>16</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 **SID[31:0]**: Size Identification
SID[31:8]: fixed code that characterizes the ADC_SIDR register. This field is always read at 0xA3C5DD.
SID[7:0]: read-only numeric field that returns the address offset (in Kbytes) of the identification registers from the IP base address:
0x01: 1 Kbytes address offset
0x02: 2 Kbytes address offset
0x04: 4 Kbytes address offset
0x08: 8 Kbytes address offset
### 26.8 ADC register map

The following table summarizes the ADC registers.

#### Table 254. ADC global register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000 - 0x0B4</td>
<td>Master ADC1</td>
</tr>
<tr>
<td>0x0B8 - 0x0FC</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x100 - 0x1B4</td>
<td>Slave ADC2</td>
</tr>
<tr>
<td>0x1B8 - 0x2FC</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x300 - 0x30C</td>
<td>Master and slave ADCs common registers</td>
</tr>
</tbody>
</table>

#### Table 255. ADC register map and reset values for each ADC (offset = 0x000 for master ADC, 0x100 for slave ADC)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>ADC_ISR</td>
<td>[reset value]</td>
</tr>
<tr>
<td>0x04</td>
<td>ADC_IER</td>
<td>[reset value]</td>
</tr>
<tr>
<td>0x08</td>
<td>ADC_CR</td>
<td>[reset value]</td>
</tr>
<tr>
<td>0x0C</td>
<td>ADC_CFGR</td>
<td>[reset value]</td>
</tr>
<tr>
<td>0x0C</td>
<td>ADC_CFGR2</td>
<td>[reset value]</td>
</tr>
<tr>
<td>0x14</td>
<td>ADC_SMPR1</td>
<td>[reset value]</td>
</tr>
<tr>
<td>0x18</td>
<td>ADC_SMPR2</td>
<td>[reset value]</td>
</tr>
<tr>
<td>0x1C</td>
<td>ADC_TR1</td>
<td>[reset value]</td>
</tr>
<tr>
<td>0x20</td>
<td>ADC_SMPR2</td>
<td>[reset value]</td>
</tr>
</tbody>
</table>

[reset values with specific values for each register]
Table 255. ADC register map and reset values for each ADC (offset = 0x000 for master ADC, 0x100 for slave ADC) (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
<th>Value (Hex)</th>
<th>Value (Dec)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x24</td>
<td>ADC_TR2</td>
<td>11111111</td>
<td>00000000</td>
<td>HT2[7:0]</td>
<td>LT2[7:0]</td>
</tr>
<tr>
<td>0x28</td>
<td>ADC_TR3</td>
<td>11111111</td>
<td>00000000</td>
<td>HT3[7:0]</td>
<td>LT3[7:0]</td>
</tr>
<tr>
<td>0x30</td>
<td>ADC_SQR1</td>
<td>00000000</td>
<td>00000000</td>
<td>SQ4[4:0]</td>
<td>SQ3[4:0]</td>
</tr>
<tr>
<td>0x34</td>
<td>ADC_SQR2</td>
<td>00000000</td>
<td>00000000</td>
<td>SQ8[4:0]</td>
<td>SQ7[4:0]</td>
</tr>
<tr>
<td>0x38</td>
<td>ADC_SQR3</td>
<td>00000000</td>
<td>00000000</td>
<td>SQ14[4:0]</td>
<td>SQ13[4:0]</td>
</tr>
<tr>
<td>0x3C</td>
<td>ADC_SQR4</td>
<td>00000000</td>
<td>00000000</td>
<td>SQ16[4:0]</td>
<td>SQ15[4:0]</td>
</tr>
<tr>
<td>0x40</td>
<td>ADC_DR</td>
<td>00000000</td>
<td>00000000</td>
<td>regular RDATA[15:0]</td>
<td></td>
</tr>
<tr>
<td>0x44-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4C</td>
<td>ADC_JSQR</td>
<td>00000000</td>
<td>00000000</td>
<td>JSQ4[4:0]</td>
<td>JSQ3[4:0]</td>
</tr>
<tr>
<td>0x50-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x60</td>
<td>ADC_OFR1</td>
<td>00000000</td>
<td>00000000</td>
<td>OFFSET[11:0]</td>
<td></td>
</tr>
<tr>
<td>0x64</td>
<td>ADC_OFR2</td>
<td>00000000</td>
<td>00000000</td>
<td>OFFSET[11:0]</td>
<td></td>
</tr>
<tr>
<td>0x68</td>
<td>ADC_OFR3</td>
<td>00000000</td>
<td>00000000</td>
<td>OFFSET[11:0]</td>
<td></td>
</tr>
<tr>
<td>0x6C</td>
<td>ADC_OFR4</td>
<td>00000000</td>
<td>00000000</td>
<td>OFFSET[11:0]</td>
<td></td>
</tr>
<tr>
<td>0x70-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Values are shown in hexadecimal and decimal format.
### Table 255. ADC register map and reset values for each ADC (offset = 0x000 for master ADC, 0x100 for slave ADC) (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x80</td>
<td>ADC_JDR1</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x84</td>
<td>ADC_JDR2</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x88</td>
<td>ADC_JDR3</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x8C</td>
<td>ADC_JDR4</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x90-0xC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xA0</td>
<td>ADC_AWD2CR</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xA4</td>
<td>ADC_AWD3CR</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xA8-0xAC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xB0</td>
<td>ADC_DIFSEL</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xB4</td>
<td>ADC_CALFACT</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xB8-0xC4</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xC0</td>
<td>ADC_OR</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xCC-0xFC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 256. ADC register map and reset values (master and slave ADC common registers)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x300</td>
<td>ADC_CSR</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x304</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x308</td>
<td>ADC_CCR</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
</tbody>
</table>

1134/3119
Table 256. ADC register map and reset values (master and slave ADC common registers) (continued)

| Offset | Register name reset value | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|----------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x30C  | ADC_CDR                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x310- 0x3EC | Reserved                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x3F0  | ADC_HWCFGR0                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                | 0x0000 1212 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x3F4  | ADC_VERR                   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                | 0x0000 0012 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x3F8  | ADC_IPDR                   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 1   | 1   | 0   |     |     |     |     |     |     |
| 0x3FC  | ADC_SIDR                   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                | 1   | 0   | 1   | 0   | 0   | 0   | 1   | 1   | 1   | 1   | 0   | 1   | 0   | 0   | 0   | 1   | 1   | 1   | 0   | 1   | 1   | 1   | 0   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 1   |

Refer to Section 2.3 on page 111 for the register boundary addresses.
27 Digital temperature sensor (DTS)

27.1 Introduction

The device embeds a sensor that converts the temperature into a square wave which frequency is proportional to the temperature. The frequency is measured either with the PCLK or the LSE clock.

27.2 DTS main features

The temperature sensor block main features are the following:

- Start of measurement triggered by software or 4 hardware sources
- Programmable sampling time to increase temperature measurement precision
- Counter synchronized on LSE or PCLK clock
- Temperature watchdog on low and high threshold
- Interrupt generation when the temperature is lower or higher than predefined thresholds and at the end of measurement.
- Asynchronous wakeup signal generation when the temperature is higher/lower than a predefined threshold (LSE mode only)
- Quick measurement using LSE clock
27.3  DTS functional description

27.3.1  DTS block diagram

The temperature sensor block diagram is shown in Figure 259.

Figure 259. Temperature sensor functional block diagram

27.3.2  DTS internal signals

Table 257. DTS internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dts_lse</td>
<td>Digital input</td>
<td>LSE clock</td>
</tr>
<tr>
<td>dts_pclk</td>
<td>Digital input</td>
<td>APB clock</td>
</tr>
<tr>
<td>dts_it</td>
<td>Digital output</td>
<td>Temperature sensor interrupt</td>
</tr>
<tr>
<td>dts_wkup</td>
<td>Digital output</td>
<td>Temperature sensor wakeup</td>
</tr>
</tbody>
</table>
27.3.3 DTS block operation

The analog part of the temperature sensor outputs a frequency that is proportional to the absolute temperature (CLK_PTAT). The frequency measurement is based on the PCLK or the LSE clock.

Before each measurement, the temperature sensor performs a calibration of the frequency generation blocks.

27.3.4 Operating modes

Several operating modes can be selected by setting the REFCLK_SEL bit in Temperature sensor configuration register 1 (DTS_CFGR1):

- PCLK only (REFCLK_SEL = 0)
  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using PCLK clock

- PCLK and LSE (REFCLK_SEL = 1)
  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using the LSE clock.

- LSE only (REFCLK_SEL = 1) and PCLK OFF
  The registers cannot be accessed. The measurement can be performed using the LSE clock. This mode is used to exit from Sleep mode by using hardware triggers and the asynchronous interrupt line.

27.3.5 Calibration

The temperature sensor must run the calibration prior to any frequency measurement. The calibration is performed automatically when the temperature measurement is triggered except for quick measurement mode (Q_MEAS_OPT set to 1 in DTS_CFGR1).

27.3.6 Prescaler

When a calibration is ongoing, the counter clock must be slower than 1 MHz. This is achieved by the PCLK clock prescaler embedded in the temperature sensor.

During the temperature measurement period, the prescaler is bypassed.

- When PCLK is used as reference clock (REFCLK_SEL set to 0 in DTS_CFGR1), a prescaler is used. Its division ratio must be configured up to 127 (refer to the HSREF_CLK_DIV[6:0] register definition for the divider setting).
- When LSE is used as reference clock (REFCLK_SEL set to 1 in DTS_CFGR1), the timebase is equal to 2 LSE periods. In this case, no prescaler is used.
27.3.7 Temperature measurement principles

The analog part of temperature sensor outputs a signal (CLK_PTAT) which FM(T) frequency is temperature-dependent.

Either PCLK or LSE can be selected as reference clock (REF_CLK) through the REFCLK_SEL bit in DTS_CFGR1.

The counting method depends on the REF_CLK frequency. This is due to the fact that two counters are implemented in the temperature sensor block:

- For low REF_CLK frequencies, a counting of FM(T) cycles is performed during one or several REF_CLK cycles.
- For high REF_CLK frequencies, a counting of REF_CLK cycles is performed during one or several FM(T) cycles.

This counter behavior is shown in *Figure 260* and *Figure 261*.

**Figure 260. Method for low REF_CLK frequencies**

![Figure 260. Method for low REF_CLK frequencies](image1)

1. To increase the precision, FM(T) measurement can be done on several LSE periods.

**Figure 261. Method for high REF_CLK frequencies**

![Figure 261. Method for high REF_CLK frequencies](image2)

1. To increase the precision, PCLK measurement can be done on several FM(T) periods.

The counting result is stored in the DTS_DR register (see *Temperature sensor data register (DTS_DR)*).

Once the FM(T) frequency has been obtained, the corresponding temperature can be calculated by software using the following formula:

- When PCLK is used:

\[
T = T_0 + \left( \frac{F_{PCLK}}{F_{PCLK}} \cdot \frac{TS1_{MFREQ}}{TS1_{SMP\_TIME}} \cdot 100 \cdot TS1_{FMT0} \right) \div TS1_{RAMP\_COEFF}
\]

where

- \( T_0 \) (factory calibration temperature) is equal to 30 °C.
- TS1_FMT0 is measured and stored in the DTS_T0VALR1 register. It is expressed in hundreds of Hertz.
TS1_RAMP_COEFF is measured during tests in factory and stored in DTS_RAMPVALR register. This value is expressed in Hz/°C.

- When the LSE clock is used

\[ T = T_0 + \left( F_{\text{LSE}} \times \text{TS1_MFREQ} / \text{TS1_SMP_TIME} - (100 \times \text{TS1_FMT0}) / \text{TS1_RAMP_COEFF} \right) \]

### 27.3.8 Sampling time

The sampling period can be increased to improve measurement accuracy. This is useful when the reference frequency (REF_CLK) is close to the FM(T) frequency. The default value is one REF_CLK cycle in LSE mode, and one FM(T) cycle in PCLK mode.

The sampling time is configured through TS1_SMP_TIME bits in DTS_CFGR1 register (see Table 258).

#### Table 258. Sampling time configuration

<table>
<thead>
<tr>
<th>TS1_SMP_TIME[3:0]</th>
<th>LSE or FM(T) clock cycle(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
</tr>
<tr>
<td>1100</td>
<td>12</td>
</tr>
<tr>
<td>1101</td>
<td>13</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
</tr>
</tbody>
</table>

### 27.3.9 Quick measurement mode

If a high precision is not required, the calibration step included in each measurement sequence can be skipped by setting Q_MEAS_OPT to 1 in the DTS_CFGR1 register. This method must used only when the LSE clock is selected as reference clock (LSREF_CLK set to 1). This mode can reduce the measurement time.
27.3.10 Trigger input

The temperature measurement can be triggered either by software or by an external event. The trigger source can be selected through TS1_INTRIG[3:0] bits in DTS_CFRG1.

<table>
<thead>
<tr>
<th>Name</th>
<th>TS1_INTRIG[3:0]</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N.A</td>
<td>0 0 0 0</td>
<td>No hardware trigger</td>
</tr>
<tr>
<td>ts1_trg0</td>
<td>0 0 0 1</td>
<td>lptim1_ch1</td>
</tr>
<tr>
<td>ts1_trg1</td>
<td>0 0 1 0</td>
<td>lptim2_ch1</td>
</tr>
<tr>
<td>ts1_trg2</td>
<td>0 0 1 1</td>
<td>lptim3_ch1</td>
</tr>
<tr>
<td>ts1_trg3</td>
<td>0 1 0 0</td>
<td>exti13</td>
</tr>
<tr>
<td>ts1_trg4</td>
<td>0 1 0 1</td>
<td></td>
</tr>
<tr>
<td>ts1_trg5</td>
<td>0 1 1 1</td>
<td></td>
</tr>
<tr>
<td>ts1_trg6</td>
<td>0 1 1 1</td>
<td></td>
</tr>
<tr>
<td>ts1_trg7</td>
<td>1 0 0 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg8</td>
<td>1 0 0 1</td>
<td></td>
</tr>
<tr>
<td>ts1_trg9</td>
<td>1 0 1 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg10</td>
<td>1 0 1 1</td>
<td></td>
</tr>
<tr>
<td>ts1_trg11</td>
<td>1 1 0 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg12</td>
<td>1 1 0 1</td>
<td></td>
</tr>
<tr>
<td>ts1_trg13</td>
<td>1 1 1 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg14</td>
<td>1 1 1 1</td>
<td></td>
</tr>
</tbody>
</table>

Note: Hardware triggers are active only on the rising edge.

The temperature sensor can only capture a hardware trigger rising edge when TS1_RDY bit is set (see Section 27.3.11: On-off control and ready flag, otherwise the trigger is ignored.

If a trigger source changes on-the-fly, the new trigger source signal should be low. If the new source signal is high, the temperature sensor detects a rising edge and start the measurement sequence.

27.3.11 On-off control and ready flag

The DTS block can be enabled by setting TS1_EN bit in DTS_CFRG1 register. The TS1_RDY flag in the Temperature sensor status register (DTS_SR) indicate that the DTS block is ready for temperature measurement: when TS1_RDY bit is set to 1, the measurement can be started. Once a measurement has started, TS1_RDY bit is reset. New measurement requests are then ignored. Once the measurement is finished, TS1_RDY bit is set again to indicate the sensor is ready to start a new measurement.
27.3.12 Temperature measurement sequence

Start of measurement can be triggered by software or hardware.

Software trigger

The software trigger is selected when TS1_INTRIG_SEL[3:0] is set to ‘0000’ in DTS_CFGR1.

If TS1_RDY is set to 1, writing TS1_START bit to 1 in DTS_CFGR1 starts the measurement.

If TS1_RDY equals 0, the software trigger does not start until TS1_RDY is set.

If TS1_START bit is kept at 1 once the measurement is finished, then the TS1_RDY flag becomes 1 and the measurement restarts.

Hardware trigger

TS1_INTRIG_SEL[3:0] bits allow selecting one hardware trigger out of 4. If TS1_RDY is set to 1, a rising edge on the trigger signal starts the measurement. When TS1_RDY is 0, the rising edge is ignored.

Temperature measurement sequence

One measurement contains two steps: the calibration of the analog blocks and the measurement. The calibration automatically starts when the measurement is triggered (see Section 27.3.5: Calibration). The measurement period depends on the following DTS_CFGR1 bits:

- the reference clock selected through REFCLK_SEL bit
- the divider ratio configured by HSREF_CLK_DIV bits
- the sampling time defined by TS1_SMP_TIME bits.

Figure 262. Temperature sensor sequence
27.4 **DTS low-power modes**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>Only works in LSE mode. DTS interrupt causes the device to exit from Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>Only works in LSE mode. DTS interrupt cause the device to exit from Stop mode.</td>
</tr>
</tbody>
</table>

**Table 260. Temperature sensor behavior in low-power modes**

27.5 **DTS interrupts**

There are two ways to use the DTS block as an interrupt source. The DTS interrupt line can be connected to the EXTI controller (see *Section 27.5.3: Asynchronous wakeup*) or to the CPU NVIC (see *Section 27.5.2: Synchronous interrupt*).

27.5.1 **Temperature window comparator**

The DTS_ITR1 register allows defining the high and low threshold that are used for temperature comparison. If the temperature data is equal or higher than TS1_HITTHD, or equal or lower than TS1_LITTHD bit, an interrupt is generated and the corresponding flag, TS1_ITLF, TS1ITHF, TS1_AITLF and TS1_AITHF, is set in the DTS_SR register (see *Section 27.6.6*).

27.5.2 **Synchronous interrupt**

A global interrupt output line is available on the DTS block. The interrupt can be generated at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see *Section 27.5.1: Temperature window comparator*).

Three interrupt events can be select via 3 bits in DTS_ITENR register (see *Section 27.6.7*). All combinations of interrupts are allowed.

The TS1_ITEF, TS1 ITLF and TS1_ITHF flags in the DTS_SR register reflect the interrupt event. They can be reset with the correspond bits of the DTS_ICIFR register (see *Section 27.6.8*).

27.5.3 **Asynchronous wakeup**

The DTS block also provides an asynchronous interrupt line. It is used only when the LSE is selected as reference clock (REFCLK_SEL=1).

This line can generate a signal that wakes up the system from Sleep mode at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see *Section 27.5.1: Temperature window comparator*).

Three asynchronous wakeup events can be selected via 3 bits in DTS_ITENR register. All combination of interrupts are allowed.
The TS1_AITEF, TS1_AITLF and TS1_AITHF flags in the DTS_SR register reflect the interrupt status. They can be reset with the correspond bits of the DTS_ICIFR register.

The following table shows the interrupt bits an the their description.

### Table 261. Interrupt control bits

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Interrupt flag</th>
<th>Enable control bit</th>
<th>Interrupt clear bit</th>
<th>Exit from Sleep mode</th>
<th>Synchronous/ Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the end of measurement</td>
<td>TS1_ITEF in DTS_SR</td>
<td>TS1_ITEEN in DTS_ITENR</td>
<td>TS1_CITEF in DTS_ICIFR</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>When the measure is equal or exceeds the low threshold</td>
<td>TS1_ITLF in DTS_SR</td>
<td>TS1_ITLEN in DTS_ITENR</td>
<td>TS1_CIITLF in DTS_ICIFR</td>
<td>NO</td>
<td>Synchronous on PCLK</td>
</tr>
<tr>
<td>When the measure is equal or exceeds the high threshold</td>
<td>TS1_IITHF in DTS_SR</td>
<td>TS1_ITIHEN in DTS_ITENR</td>
<td>TS1_CITHF in DTS_ICIFR</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>At the end of measurement</td>
<td>TS1_AITEF in DTS_SR</td>
<td>TS1_AITEEN in DTS_ITENR</td>
<td>TS1_CAITEF in DTS_ICIFR</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>When the measure is equal or exceeds the low threshold</td>
<td>TS1_AITLF in DTS_SR</td>
<td>TS1_AITLEN in DTS_ITENR</td>
<td>TS1_CAITLF in DTS_ICIFR</td>
<td>YES</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>When the measure is equal or exceeds the high threshold</td>
<td>TS1_AITHF in DTS_SR</td>
<td>TS1_AITHEN in DTS_ITENR</td>
<td>TS1_CAITHF in DTS_ICIFR</td>
<td>YES</td>
<td></td>
</tr>
</tbody>
</table>

### 27.6 DTS registers

The registers of this peripheral can only be accessed by-word (32-bit).

#### 27.6.1 Temperature sensor configuration register 1 (DTS_CFRG1)

DTS_CFRG1 is the configuration register for temperature sensor 1.

Address offset: 0x00
Reset value: 0x0000 0000
Bit 31  Reserved, must be kept at reset value.

Bits 30:24  **HSREF_CLK_DIV[6:0]**: High speed clock division ratio
These bits are set and cleared by software. They can be used to define the division ratio for the main clock in order to obtain the internal frequency lower than 1 MHz required for the calibration. They are applicable only for calibration when PCLK is selected as reference clock (REFCLK_SEL=0).
- 0000000: No divider
- 0000001: No divider
- 0000010: 1/2 division ratio
- ...
- 1111111: 1/127 division ratio

Bits 23:22  Reserved, must be kept at reset value.

Bit 21  **Q_MEAS_OPT**: Quick measurement option bit
This bit is set and cleared by software. It is used to increase the measurement speed by suppressing the calibration step. It is effective only when the LSE clock is used as reference clock (REFCLK_SEL=1).
- 0: Measurement with calibration
- 1: Measurement without calibration

Bit 20  **REFCLK_SEL**: Reference clock selection bit
This bit is set and cleared by software. It indicates whether the reference clock is the high speed clock (PCLK) or the low speed clock (LSE).
- 0: High speed reference clock (PCLK)
- 1: Low speed reference clock (LSE)

Bits 19:16  **TS1_SMP_TIME[3:0]**: Sampling time for temperature sensor 1
These bits allow increasing the sampling time to improve measurement precision.
When the PCLK clock is selected as reference clock (REFCLK_SEL = 0), the measurement is performed at TS1_SMP_TIME period of CLK_PTAT.
When the LSE is selected as reference clock (REFCLK_SEL = 1), the measurement is performed at TS1_SMP_TIME period of LSE.

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

Bits 11:8  **TS1_INTRIG_SEL[3:0]**: Input trigger selection bit for temperature sensor 1
These bits are set and cleared by software. They select which input triggers a temperature measurement. Refer to *Section 27.3.10: Trigger input*.

Bits 7:5  Reserved, must be kept at reset value.

Bit 4  **TS1_START**: Start frequency measurement on temperature sensor 1
This bit is set and cleared by software.
- 0: No software trigger.
- 1: Software trigger for a frequency measurement. (only if TS1 is ready).

Bits 3:1  Reserved, must be kept at reset value.

Bit 0  **TS1_EN**: Temperature sensor 1 enable bit
This bit is set and cleared by software.
- 0: Temperature sensor 1 disabled
- 1: Temperature sensor 1 enabled

*Note: Once enabled, the temperature sensor is active after a specific delay time. The TS1_RDY flag is set when the sensor is ready.*
27.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1)

DTS_T0VALR1 contains the value of the factory calibration temperature (T0) for temperature sensor 1. The reset value is factory trimmed.

Address offset: 0x08
Reset value: 0x000X XXXX

---

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
<th>TS1_T0[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 17:16 **TS1_T0[1:0]**: Engineering value of the T0 temperature for temperature sensor 1.
- 00: 30 °C
- 01: 130 °C
- Others: Reserved, must not be used.

Bits 15:0 **TS1_FMT0[15:0]**: Engineering value of the frequency measured at T0 for temperature sensor 1.
This value is expressed in 0.1 kHz.

---

27.6.3 Temperature sensor ramp value register (DTS_RAMPVALR)

The DTS_RAMPVALR is the ramp coefficient for the temperature sensor. The reset value is factory trimmed.

Address offset: 0x10
Reset value: 0xXXXX XXXX

---

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
<th>TS1_RAMP_COEFF[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TS1_RAMP_COEFF[15:0]**: Engineering value of the ramp coefficient for the temperature sensor 1.
This value is expressed in Hz/°C.
### 27.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1)

DTS_ITR1 contains the threshold values for sensor 1.

Address offset: 0x14  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS1_HITTHD[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16 **TS1_HITTHD[15:0]**: High interrupt threshold for temperature sensor 1  
These bits are set and cleared by software. They indicate the highest value than can be reached before raising an interrupt signal.

Bits 15:0 **TS1_LITTHD[15:0]**: Low interrupt threshold for temperature sensor 1  
These bits are set and cleared by software. They indicate the lowest value than can be reached before raising an interrupt signal.

### 27.6.5 Temperature sensor data register (DTS_DR)

The DTS_DR contains the number of REF_CLK cycles used to compute the FM(T) frequency.

Address offset: 0x1C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS1_MFREQ[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16 **Reserved, must be kept at reset value.**

Bits 15:0 **TS1_MFREQ[15:0]**: Value of the counter output value for temperature sensor 1
### 27.6.6 Temperature sensor status register (DTS_SR)

Address offset: 0x20  
Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS1_RDY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

- **Bit 15** **TS1_RDY**: Temperature sensor 1 ready flag
  - This bit is set and reset by hardware.
  - It indicates that a measurement is ongoing.
  - 0: Temperature sensor 1 busy
  - 1: Temperature sensor 1 ready

Bits 14:7 Reserved, must be kept at reset value.

- **Bit 6** **TS1_AITHF**: Asynchronous interrupt flag for high threshold on temperature sensor 1
  - This bit is set by hardware when the high threshold is reached.
  - It is cleared by software by writing 1 to the TS1_CAITHF bit in the DTS_ICIFR register.
  - 0: High threshold not reached on temperature sensor 1
  - 1: High threshold reached on temperature sensor 1
  - *Note: This bit is active only when the TS1_AITHFEN bit is set*

- **Bit 5** **TS1_AITLF**: Asynchronous interrupt flag for low threshold on temperature sensor 1
  - This bit is set by hardware when the low threshold is reached.
  - It is cleared by software by writing 1 to the TS1_CAITLF bit in the DTS_ICIFR register.
  - 0: Low threshold not reached on temperature sensor 1
  - 1: Low threshold reached on temperature sensor 1
  - *Note: This bit is active only when the TS1_AITLFEN bit is set*

- **Bit 4** **TS1_AITEF**: Asynchronous interrupt flag for end of measure on temperature sensor 1
  - This bit is set by hardware when a temperature measure is done.
  - It is cleared by software by writing 1 to the TS1_CAITEF bit in the DTS_ICIFR register.
  - 0: End of measure not detected on temperature sensor 1
  - 1: End of measure detected on temperature sensor 1
  - *Note: This bit is active only when the TS1_AITEFEN bit is set*

Bits 3 Reserved, must be kept at reset value.
Bit 2 **TS1ITHF**: Interrupt flag for high threshold on temperature sensor 1, synchronized on PCLK.
   This bit is set by hardware when the high threshold is set and reached.
   It is cleared by software by writing 1 to the TS1_CITHF bit in the DTS_ICIFR register.
   0: High threshold not reached on temperature sensor 1
   1: High threshold reached on temperature sensor 1
   *Note: This bit is active only when the TS1ITHFEN bit is set*

Bit 1 **TS1ITLF**: Interrupt flag for low threshold on temperature sensor 1, synchronized on PCLK.
   This bit is set by hardware when the low threshold is set and reached.
   It is cleared by software by writing 1 to the TS1_CITLF bit in the DTS_ICIFR register.
   0: Low threshold not reached on temperature sensor 1
   1: Low threshold reached on temperature sensor 1
   *Note: This bit is active only when the TS1ITLFEN bit is set*

Bit 0 **TS1ITEF**: Interrupt flag for end of measurement on temperature sensor 1, synchronized on PCLK.
   This bit is set by hardware when a temperature measure is done.
   It is cleared by software by writing 1 to the TS2_CITEF bit in the DTS_ICIFR register.
   0: No end of measurement detected on temperature sensor 1
   1: End of measure detected on temperature sensor 1
   *Note: This bit is active only when the TS1ITEFEN bit is set*

27.6.7 Temperature sensor interrupt enable register (DTS_ITENR)

Address offset: 0x24
Reset value: 0x0000 0000

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1AITHEN**: Asynchronous interrupt enable flag on high threshold for temperature sensor 1.
   This bit are set and cleared by software.
   It enables the asynchronous interrupt when the temperature is above the high threshold
   (only when REFCLK_SEL = 1’b1)
   0: Asynchronous interrupt on high threshold disabled for temperature sensor 1
   1: Asynchronous interrupt on high threshold enabled for temperature sensor 1

Bit 5 **TS1AITLEN**: Asynchronous interrupt enable flag for low threshold on temperature sensor 1.
   This bit are set and cleared by software.
   It enables the asynchronous interrupt when the temperature is below the low threshold (only
   when REFCLK_SEL = 1)
   0: Asynchronous interrupt on low threshold disabled for temperature sensor 1
   1: Asynchronous interrupt on low threshold enabled for temperature sensor 1
Digital temperature sensor (DTS)  RM0481

27.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR)

DTS_ICIFR is the control register for the interrupt flags.

Address offset: 0x28

Reset value: 0x0000 0000

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

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

| rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1_CAITHF**: Asynchronous interrupt clear flag for high threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_AITHF flag in the DTS_SR register.

Bit 5 **TS1_CAITLF**: Asynchronous interrupt clear flag for low threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_AITLF flag in the DTS_SR register.
Bit 4 **TS1_CAITEF**: Write once bit. Clear the asynchronous IT flag for End Of Measure for thermal sensor 1.
   Writing 1 clears the TS1_AITEF flag of the DTS_SR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_CITHF**: Interrupt clear flag for high threshold on temperature sensor 1
   Writing this bit to 1 clears the TS1_ITHF flag in the DTS_SR register.

Bit 1 **TS1_CITLF**: Interrupt clear flag for low threshold on temperature sensor 1
   Writing 1 to this bit clears the TS1_ITLF flag in the DTS_SR register.

Bit 0 **TS1_CITEF**: Interrupt clear flag for end of measurement on temperature sensor 1
   Writing 1 to this bit clears the TS1_ITEF flag in the DTS_SR register.

### 27.6.9 Temperature sensor option register (DTS_OR)

The DTS_OR contains general-purpose option bits.

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bits 31:0 **TS_OP[31:0]**: general purpose option bits
**27.6.10 DTS register map**

The following table summarizes the temperature sensor registers.

Refer to *Section 2.3 on page 111* for the register boundary addresses.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register reset value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>DTS_CFR1</td>
<td>0x00</td>
<td>HSREF_CLK_DIV [6:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00</td>
<td>O_MEAS.CODE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00</td>
<td>REF_CLK[3:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00</td>
<td>TS1_SMP_TIME [3:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00</td>
<td>TS1_INTRIG_SEL [3:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00</td>
<td>TS1.START</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00</td>
<td>TS1.INTRIG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00</td>
<td>TS1.EN</td>
</tr>
<tr>
<td>0x04</td>
<td>DTS_T0VALR1</td>
<td>0x00</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x08</td>
<td>DTS_T0VALR1</td>
<td>0x00</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0C</td>
<td>DTS_RAMPVALR</td>
<td>0x00</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x10</td>
<td>DTS_RAMPVALR</td>
<td>0x00</td>
<td>TS1_RAMP_COEFF[15:0]</td>
</tr>
<tr>
<td>0x14</td>
<td>DTS_ITR1</td>
<td>0x00</td>
<td>TS1_HITTHD[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00</td>
<td>TS1_LITTHD[15:0]</td>
</tr>
<tr>
<td>0x18</td>
<td>DTS_DR</td>
<td>0x00</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x1C</td>
<td>DTS_DR</td>
<td>0x00</td>
<td>TS1_MFREQ[15:0]</td>
</tr>
<tr>
<td>0x20</td>
<td>DTS_SR</td>
<td>0x00</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x24</td>
<td>DTS_ITENR</td>
<td>0x00</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x28</td>
<td>DTS_ICIFR</td>
<td>0x00</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x2C</td>
<td>DTS_OR</td>
<td>0x00</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Refer to *Section 2.3 on page 111* for the register boundary addresses.
28 Digital-to-analog converter (DAC)

28.1 Introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data can be left- or right-aligned. The DAC features two output channels, each with its own converter. In dual DAC channel mode, conversions can be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, \( V_{\text{REF}+} \) (shared with others analog peripherals) is available for better resolution. An internal reference can also be set on the same input. Refer to voltage reference buffer (VREFBUF) section.

The DAC output buffer can be optionally enabled to obtain a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the Sample and hold mode.

28.2 DAC main features

The DAC main features are the following (see Figure 263: Dual-channel DAC block diagram)

- One DAC interface, maximum two output channels
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- Double data DMA capability to reduce the bus activity
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Sample and hold mode for low power operation in Stop mode
- Input voltage reference from \( V_{\text{REF}+} \) pin or internal VREFBUF reference

Figure 263 shows the block diagram of a DAC channel and Table 264 gives the pin description.
### 28.3 DAC implementation

Table 263. DAC features

<table>
<thead>
<tr>
<th>DAC features</th>
<th>DAC1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual channel</td>
<td>X</td>
</tr>
<tr>
<td>Output buffer</td>
<td>X</td>
</tr>
<tr>
<td>I/O connection</td>
<td>DAC1_OUT1 on PA4, DAC1_OUT2 on PA5</td>
</tr>
<tr>
<td>Maximum sampling time</td>
<td>1 Msps</td>
</tr>
<tr>
<td>Autonomous mode</td>
<td>-</td>
</tr>
<tr>
<td>VREF+ pin</td>
<td>X</td>
</tr>
</tbody>
</table>
28.4 DAC functional description

28.4.1 DAC block diagram

Figure 263. Dual-channel DAC block diagram

1. MODEx bits in the DAC_MCR control the output mode and allow switching between the Normal mode in buffer/unbuffered configuration and the Sample and hold mode.

2. Refer to Section 28.3: DAC implementation for channel2 availability.
28.4.2 DAC pins and internal signals

The DAC includes:
- Up to two output channels
- DAC output channel buffered or non-buffered
- Sample and hold block and registers operational in Stop mode, using the LSI/LSE clock source (dac_hold_ck) for static conversion.

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DACx_OUTy output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The Sample and hold block and its associated registers can run in Stop mode using the LSI/LSE clock source (dac_hold_ck).

### Table 264. DAC input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>VREF+</td>
<td>Input, analog reference positive</td>
<td>The higher/positive reference voltage for the DAC, VREF+ ≤ VDDAmax (refer to datasheet)</td>
</tr>
<tr>
<td>VDDA</td>
<td>Input, analog supply</td>
<td>Analog power supply</td>
</tr>
<tr>
<td>VSSA</td>
<td>Input, analog supply ground</td>
<td>Ground for analog power supply</td>
</tr>
<tr>
<td>DACx_OUTy</td>
<td>Analog output signal</td>
<td>DACx channel analog output</td>
</tr>
</tbody>
</table>

### Table 265. DAC internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dac_ch1_dma</td>
<td>Bidirectional</td>
<td>DAC channel1 DMA request/acknowledge</td>
</tr>
<tr>
<td>dac_ch2_dma</td>
<td>Bidirectional</td>
<td>DAC channel2 DMA request/acknowledge</td>
</tr>
<tr>
<td>dac_ch1_trgx (x = 1 to 15)</td>
<td>Inputs</td>
<td>DAC channel1 trigger inputs</td>
</tr>
<tr>
<td>dac_ch2_trgx (x = 1 to 15)</td>
<td>Inputs</td>
<td>DAC channel2 trigger inputs</td>
</tr>
<tr>
<td>dac_unr_it</td>
<td>Output</td>
<td>DAC underrun interrupt</td>
</tr>
<tr>
<td>dac_hclk</td>
<td>Input</td>
<td>DAC peripheral clock</td>
</tr>
<tr>
<td>dac_ker_ck</td>
<td>Input</td>
<td>DAC kernel clock</td>
</tr>
<tr>
<td>dac_hold_ck</td>
<td>Input</td>
<td>DAC low-power clock used in Sample and hold mode</td>
</tr>
</tbody>
</table>

### Table 266. DAC interconnection

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source</th>
<th>Source type</th>
</tr>
</thead>
<tbody>
<tr>
<td>dac_hold_ck</td>
<td>ck_lsi or ck_lse</td>
<td>LSI or LSE clock selected in the RCC</td>
</tr>
<tr>
<td>dac_chx_trg1 (x = 1, 2)</td>
<td>tim1_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg2 (x = 1, 2)</td>
<td>tim2_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg3 (x = 1, 2)</td>
<td>tim4_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg4 (x = 1, 2)</td>
<td>tim5_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg5 (x = 1, 2)</td>
<td>tim6_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
</tbody>
</table>
28.4.3 DAC clocks

Two clock sources can be used to update the DAC:
- `dac_hclk`: DAC peripheral clock (AHB clock)
- `dac_ker_ck`: DAC kernel clock: this clock can be used to synchronize DAC and ADC.
- `dac_hold_ck`: low-power clock used in Sample and hold mode

The DAC clock is selected in the RCC.

28.4.4 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a tWAKEUP startup time.

DACxRDY bit is set in the DAC_SR register when the DAC interface is ready to accept data. Writing new data or asserting the trigger is not allowed when ENx bit is set while DACxRDY signal is reset.

*Note:* The ENx bit enables the analog DAC channelx only. The DAC channelx digital interface is enabled even if the ENx bit is reset.

28.4.5 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:
- **Single DAC channel**
  - There are three possibilities:
    - 8-bit right alignment: the software has to load data into the DAC_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)
    - 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)
    - 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyxy register, the data written by the user is shifted and stored into the corresponding DHRx (data holding registerx, which are internal non-memory-mapped registers). The DHRx register is then loaded into the DORx register either automatically, by software trigger or by an external event trigger.

**Table 266. DAC interconnection (continued)**

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source</th>
<th>Source type</th>
</tr>
</thead>
<tbody>
<tr>
<td>dac_chx_trg6 (x = 1, 2)</td>
<td>tim7_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg7 (x = 1, 2)</td>
<td>tim8_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg8 (x = 1, 2)</td>
<td>tim15_trgo</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg11 (x = 1, 2)</td>
<td>lptim1_ch1</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg12 (x = 1, 2)</td>
<td>lptim2_ch1</td>
<td>Internal signal from on-chip timers</td>
</tr>
<tr>
<td>dac_chx_trg13 (x = 1, 2)</td>
<td>exti9</td>
<td>External pin</td>
</tr>
</tbody>
</table>
Dual DAC channels (when available)

There are three possibilities:

- 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)

- 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)

- 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12RD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DAC_DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Signed/unsigned data

DAC input data are unsigned: 0x000 corresponds to the minimum value and 0xFFF to the maximum value for 12-bit mode.

The DAC can also handle signed input data in 2’s complement format. This is done by setting SINFORMATx bit in the DAC_MCR register.

When SINFORMATx bit is set, the MSB bit of the data written to DHRx registers is inverted when it is copied to the DAC_DORx register, and the DAC interface can accept signed data (Q1.15, Q1.11 or Q1.7 format). DAC_DHR12Lx register can be used to store 16-bit signed data in the data holding registers. The 12 MSBs of 16-bit data are used for the DAC output data and the MSB bit is inverted. The four LSBs are simply ignored.
28.4.6 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channel must be performed by loading the DAC_DHRx register (write operation to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12RD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one dac_hclk clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_hclk clock cycles after the trigger signal.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time \( t_{SETTLING} \) that depends on the power supply voltage and the analog output load.

To synchronize DAC and ADC, the same clock source can be used for both peripherals. This is done by selecting the dac_ker_ck clock instead of the dac_hclk clock (AHB clock) in the RCC.

HFSEL bits of DAC_MCR must be set when dac_hclk or dac_ker_ck clock speed is faster than 80 MHz. It adds an extra delay to the transfer from DAC_DHRx register to DAC_DORx register.

Refer to Table HFSEL description below for the limitation of the DAC_DORx update rate depending on HFSEL bits and dac_hclk clock frequency.

If the data is updated or a software/hardware trigger event occurs during the non-allowed period, the peripheral behavior is unpredictable.

The above timing is only related to the limitation of the DAC interface. Refer also to the \( t_{SETTLING} \) parameter value in the product datasheet.

<table>
<thead>
<tr>
<th>SINFORMATx bit</th>
<th>DATA written to DHRx register</th>
<th>DATA transfered to DORx register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x000</td>
<td>0x000</td>
</tr>
<tr>
<td>0</td>
<td>0xFFF</td>
<td>0xFFF</td>
</tr>
<tr>
<td>1</td>
<td>0x7FF</td>
<td>0xFFF</td>
</tr>
<tr>
<td>1</td>
<td>0x000</td>
<td>0x800</td>
</tr>
<tr>
<td>1</td>
<td>0xFFF</td>
<td>0x7FF</td>
</tr>
<tr>
<td>1</td>
<td>0x800</td>
<td>0x000</td>
</tr>
</tbody>
</table>
Table 268. HFSEL description

<table>
<thead>
<tr>
<th>HFSEL [1:0]</th>
<th>AHB frequency</th>
<th>Latency using AHB clock (dac_hclk)</th>
<th>Latency using dac_ker_ck clock</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>&lt; 80 MHz</td>
<td>3</td>
<td>4</td>
<td>DAC_DOR update rate up to 3 AHB clock cycles or 4 dac_ker_ck cycles.</td>
</tr>
<tr>
<td>01</td>
<td>≥ 80 MHz(^{(1)})</td>
<td>5</td>
<td>5</td>
<td>DAC_DOR update rate up to 5 AHB clock or dac_ker_ck cycles.</td>
</tr>
<tr>
<td>10</td>
<td>≥ 160 MHz</td>
<td>7</td>
<td>6</td>
<td>DAC_DOR update rate up to 7 AHB clock cycles or 6 dac_ker_ck cycles.</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1. Refer to the device datasheet for the value of the maximum AHB frequency.

**Figure 266. Timing diagram for conversion with trigger disabled TEN = 0**

![Timing diagram](image)

**28.4.7 DAC output voltage**

Digital inputs are converted to output voltages on a linear conversion between 0 and V\(\text{REF}^+\).

The analog output voltages on each DAC channel pin are determined by the following equation:

\[
\text{DACoutput} = \frac{\text{V}_{\text{REF}} \times \text{DOR}}{4096}
\]

**28.4.8 DAC trigger selection**

If the TENx control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[3:0] control bits determine which out of 16 possible events triggers the conversion as shown in TSELx[3:0] bits of the DAC_CR register. These events can be either the software trigger or hardware triggers. Refer to the interconnection table in Section 28.4.2: DAC pins and internal signals.

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC_DHRx register are transferred into the
DAC_DORx register. The DAC_DORx register is updated three dac_hclk cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

**Note:**

TSELx[3:0] bit cannot be changed when the ENx bit is set.

When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one dac_hclk clock cycle.

### 28.4.9 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC_DHRx register is transferred into the DAC_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit must be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC_DHRx to DAC_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHRx before the first trigger event occurs.

#### DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

The software must clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion can be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

#### DMA Double data mode

When the DMA controller is used in Normal mode, only 12-bit (or 8-bit) data are transferred by a DMA request. As the AHB width is 32 bits, two 12-bit data may be transferred simultaneously. To use this mode, set the DMADOUBLEx bit of DAC_MCR register.

A DAC DMA request is generated every two external triggers (except for software triggers) when the DMAENx bit is set:

1. When the first trigger is detected, the value of the DAC_DHRx and DAC_DHRBx registers are transferred into the DAC_DORx and DAC_DORBx registers. The actual
DAC data is loaded into the DAC_DORx register. A DMA request is then generated. The DMA writes the new data to the DAC_DHRx and DAC_DHRBx data registers.

2. When the next trigger is detected, the actual DAC data is loaded into the DAC_DHRBx register. This second trigger does not generate any DMA request. The DORSTATx bit indicates which DOR data is actually loaded into the analog DAC input.

DMA underrun function is also supported in DMA Double data mode.

In DMA Double mode, DMA requests can only handle one DAC channel. To use two channel outputs in DMA Double mode, each DMA channel has to be configured separately.

The following conditions must be met to change from Double data to single data mode or vice versa:
- The DAC must be disabled.
- DMAEN bit must be cleared (ENx = 0 and DMAEN = 0).

### 28.4.10 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to 01. The preloaded value in LFSR is 0xAAA. This register is updated three dac_hclk clock cycles after each trigger event, following a specific calculation algorithm.

The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then transferred into the DAC_DORx register.

If LFSR is 0x0000, a ‘1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.
Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

28.4.11 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to 10. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_hclk clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is transferred into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.
28.4.12 DAC channel modes

Each DAC channel can be configured in Normal mode or Sample and hold mode. The output buffer can be enabled to obtain a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In Normal mode, there are four combinations, by changing the buffer state and by changing the DACx_OUTy pin interconnections.

To enable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:
- 000: DAC is connected to the external pin

To disable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:
- 010: DAC is connected to the external pin

Sample and hold mode

In Sample and hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the LSI/LSE low-speed clock (dac_hold_ck) in addition to the dac_hclk clock, allowing using the DAC channels in deep low power modes such as Stop mode.

The LSI/LSE low-speed clock (dac_hold_ck) must not be stopped when the Sample and hold mode is enabled.
The sample/hold mode operations can be divided into 3 phases:

1. **Sample phase:** the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLEx[9:0] bits in DAC_SHSRx register. During the write of the TSAMPLEx[9:0] bits, the BWSTx bit in DAC_SR register is set to 1 to synchronize between both clocks domains (AHB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation.

2. **Hold phase:** the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLDx[9:0] bits in DAC_SHHR register.

3. **Refresh phase:** the refresh time is configured with the TREFRESHx[7:0] bits in DAC_SHRR register.

The timings for the three phases above are in units of LSI/LSE clock periods. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI/LSE ~32 KHz is selected:

- 12 cycles are required for sample phase: TSAMPLEx[9:0] = 11,
- 62 cycles are required for hold phase: THOLDx[9:0] = 62,
- and 4 cycles are required for refresh period: TREFRESHx[7:0] = 4.

In this example, the power consumption is reduced by almost a factor of 15 versus Normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

### Table 269. Sample and refresh timings

<table>
<thead>
<tr>
<th>Buffer State</th>
<th>$t_{SAMP}$</th>
<th>$t_{REFRESH}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>$7 \mu s + (10*R_{BON}*C_{SH})$</td>
<td>$7 \mu s + (R_{BON}<em>C_{SH})</em>\ln(2*N_{LSB})$</td>
</tr>
<tr>
<td>Disable</td>
<td>$3 \mu s + (10*R_{BOFF}*C_{SH})$</td>
<td>$3 \mu s + (R_{BOFF}<em>C_{SH})</em>\ln(2*N_{LSB})$</td>
</tr>
</tbody>
</table>

1. In the above formula the settling to the desired code value with $\frac{1}{2}$ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8 bits resolution, the settling time is 7 constant time.

2. $C_{SH}$ is the capacitor in Sample and hold mode.

3. The tolerated voltage drop during the hold phase “Vd” is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with $\frac{1}{2}$ LSB error accuracy requires $\ln(2*N_{LSB})$ constant time of the DAC.

**Example of the sample and refresh time calculation with output buffer on**

The values used in the example below are provided as indication only. Refer to the product datasheet for product data.

- $C_{SH} = 100 \text{ nF}$
- $V_{REF+} = 3.0 \text{ V}$

Sampling phase:

$t_{SAMP} = 7 \mu s + (10 * 2000 * 100 * 10^{-9}) = 2.007 \text{ ms}$

(where $R_{BON} = 2 \text{ k\Omega}$)
Refresh phase:
\[ t_{\text{REFRESH}} = 7 \ \mu s + (2000 \cdot 100 \cdot 10^{-9}) \cdot \ln(2 \cdot 10) = 606.1 \ \mu s \]
(where \( N_{\text{LSB}} = 10 \) (10 LSB drop during the hold phase))

Hold phase:
\[ D_v = i_{\text{leak}} \cdot t_{\text{hold}} / C_{\text{SH}} = 0.0073 \ \text{V} \ (10 \ \text{LSB of 12bit at 3 \ \text{V}}) \]
\[ i_{\text{leak}} = 150 \ \text{nA} \ (\text{worst case on the IO leakage on all the temperature range}) \]
\[ t_{\text{hold}} = 0.0073 \cdot 100 \cdot 10^{-9} / (150 \cdot 10^{-9}) = 4.867 \ \text{ms} \]

Figure 271. DAC Sample and hold mode phase diagram

Like in Normal mode, the Sample and hold mode has different configurations.

To enable the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:
- 100: DAC is connected to the external pin

To disabled the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:
- 110: DAC is connected to external pin

When MODEx[2:0] bits are equal to 111, an internal capacitor, \( C_{\text{Link}} \) holds the voltage output of the DAC core and then drive it to on-chip peripherals.

All Sample and hold phases are interruptible, and any change in DAC_DHRx immediately triggers a new sample phase.
The transfer function for an N-bit digital-to-analog converter (DAC) is:

\[ V_{\text{out}} = \left( \frac{D}{2^N} \right) \times G \times V_{\text{ref}} \times + V_{\text{OS}} \]

Where \( V_{\text{out}} \) is the analog output, \( D \) is the digital input, \( G \) is the gain, \( V_{\text{ref}} \) is the nominal full-scale voltage, and \( V_{\text{OS}} \) is the offset voltage. For an ideal DAC channel, \( G = 1 \) and \( V_{\text{OS}} = 0 \).

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the \( V_{\text{OS}} \), a calibration is required by a trimming technique.

The calibration is only valid when the DAC channelx is operating with buffer enabled (MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101). If applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output is disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer acts as a comparator to sense the middle-code value 0x800 and compare it to \( V_{\text{REF+}}/2 \) signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAGx bit).

Two calibration techniques are provided:

- **Factory trimming (default setting)**
  The DAC buffer offset is factory trimmed. The default value of OTRIMx[4:0] bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.

- **User trimming**
  The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when \( V_{\text{DDA}} \) voltage, temperature, \( V_{\text{REF+}} \) values change and can be done at any point during application by software.

**Note:** Refer to the datasheet for more details of the Nominal factory trimming conditions

In addition, when \( V_{\text{DD}} \) is removed (example the device enters in STANDBY or VBAT modes) the calibration is required.

The steps to perform a user trimming calibration are as below:

<table>
<thead>
<tr>
<th>MODEx[2:0]</th>
<th>Mode</th>
<th>Buffer</th>
<th>Output connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>Normal mode</td>
<td>Enabled</td>
<td>Connected to external pin</td>
</tr>
<tr>
<td>0 1 0</td>
<td>Sample and hold mode</td>
<td>Disabled</td>
<td>Connected to external pin</td>
</tr>
<tr>
<td>1 0 0</td>
<td>Normal mode</td>
<td>Enabled</td>
<td>Connected to external pin</td>
</tr>
<tr>
<td>1 1 0</td>
<td>Sample and hold mode</td>
<td>Disabled</td>
<td>Connected to external pin and to on chip peripherals (such as comparators)</td>
</tr>
</tbody>
</table>
1. If the DAC channel is active, write 0 to ENx bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101.
3. Start the DAC channel calibration, by setting the CENx bit in DAC_CR register to 1.
4. Apply a trimming algorithm:
   a) Write a code into OTRIMx[4:0] bits, starting by 0b00000.
   b) Wait for tTRIM delay.
   c) Check if CAL_FLAGx bit in DAC_SR is set to 1.
   d) If CAL_FLAGx is set to 1, the OTRIMx[4:0] trimming code is found and can be used during device operation to compensate the output value, else increment OTRIMx[4:0] and repeat sub-steps from (a) to (d) again.

   The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIMx[4:0] bits in a faster way.

   The commutation/toggle of CAL_FLAGx bit indicates that the offset is correctly compensated and the corresponding trim code must be kept in the OTRIMx[4:0] bits in DAC_CCR register.

   **Note:**
   A tTRIM delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL_FLAGx bit in DAC_SR register in order to get a correct value. This parameter is specified into datasheet electrical characteristics section.

   If VDDA, VREF+ and temperature conditions do not change during device operation while it enters more often in standby and VBAT mode, the software may store the OTRIMx[4:0] bits found in the first user calibration in the flash or in back-up registers, then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.

   **When CENx bit is set, it is not allowed to set ENx bit.**

28.4.14 Dual DAC channel conversion modes (if dual channels are available)

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to DHRxxxD registers are required. As a result, two output channels can be used either independently or simultaneously.

15 conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DHRx registers if needed.

All modes are described in the paragraphs below.

**Independent trigger without wave generation**

To configure the DAC in this conversion mode, the following sequence is required:
1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).
When a DAC channel 1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_hclk clock cycles later).

When a DAC channel 2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three dac_hclk clock cycles later).

**Independent trigger with single LFSR generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVE[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel 1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel 2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.

**Independent trigger with different LFSR generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVE[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel 1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel 2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.
**Independent trigger with single triangle generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

**Independent trigger with different triangle generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

**Simultaneous software start**

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

In this configuration, one dac_hclk clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

**Simultaneous trigger without wave generation**

To configure the DAC in this conversion mode, the following sequence is required:
1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three dac_hclk clock cycles).

**Simultaneous trigger with single LFSR generation**

1. To configure the DAC in this conversion mode, the following sequence is required:
2. Set the two DAC channel trigger enable bits TEN1 and TEN2.
3. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
4. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
5. Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

**Simultaneous trigger with different LFSR generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

**Simultaneous trigger with single triangle generation**

To configure the DAC in this conversion mode, the following sequence is required:
1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

**Simultaneous trigger with different triangle generation**

To configure the DAC in this conversion mode, the following sequence is required:
1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three AHB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the DAC channel2 triangle counter is updated.

### 28.5 DAC in low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect, DAC used with DMA.</td>
</tr>
<tr>
<td>Stop(^{(1)})</td>
<td>The DAC remains active with a static value. The Sample and hold mode can be selected using LSE/LSI clock.</td>
</tr>
<tr>
<td>Standby</td>
<td>The DAC peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

---

1. Refer to Section 28.3: DAC implementation for information on the Stop modes supported by the DAC peripheral.
# 28.6 DAC interrupts

## Table 272. DAC interrupts

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit Sleep mode</th>
<th>Exit Stop mode</th>
<th>Exit Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAC</td>
<td>DMA underrun</td>
<td>DMAUDRx</td>
<td>DMAUDRI Ex</td>
<td>Write DMAUDRx = 1</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
28.7 DAC registers

Refer to Section 1 on page 101 for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

28.7.1 DAC control register (DAC_CR)

Address offset: 0x00
Reset value: 0x0000 0000

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CEN2**: DAC channel2 calibration enable

This bit is set and cleared by software to enable/disable DAC channel2 calibration, it can be written only if EN2 bit is set to 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel2 in Normal operating mode
1: DAC channel2 in calibration mode

*Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.*

Bit 29 **DMAUDR1E2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled
1: DAC channel2 DMA underrun interrupt enabled

*Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.*

Bit 28 **DMAEN2**: DAC channel2 DMA enable

This bit is set and cleared by software.

0: DAC channel2 DMA mode disabled
1: DAC channel2 DMA mode enabled

*Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.*
Bits 27:24  **MAMP2[3:0]**: DAC channel2 mask/amplitude selector  
These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.  
0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1  
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3  
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7  
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15  
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31  
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63  
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127  
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255  
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511  
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023  
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047  
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095  

*Note:* These bits are available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bits 23:22  **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable  
These bits are set/reset by software.  
00: wave generation disabled  
01: Noise wave generation enabled  
1x: Triangle wave generation enabled  

*Note:* Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)  
These bits are available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bits 21:18  **TSEL2[3:0]**: DAC channel2 trigger selection  
These bits select the external event used to trigger DAC channel2  
0000: SWTRIG2  
0001: dac_ch2_trg1  
0010: dac_ch2_trg2  
...  
1111: dac_ch2_trg15  
Refer to the trigger selection tables in Section 28.4.2: DAC pins and internal signals for details on trigger configuration and mapping.  

*Note:* Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).  
These bits are available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 17  **TEN2**: DAC channel2 trigger enable  
This bit is set and cleared by software to enable/disable DAC channel2 trigger  
0: DAC channel2 trigger disabled and data written into the DAC_DHR2 register are transferred one dac_hclk clock cycle later to the DAC_DOR2 register  
1: DAC channel2 trigger enabled and data from the DAC_DHR2 register are transferred three dac_hclk clock cycles later to the DAC_DOR2 register  

*Note:* When software trigger is selected, the transfer from the DAC_DHR2 register to the DAC_DOR2 register takes only one dac_hclk clock cycle.  
These bits are available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.
Bit 16  **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.
- 0: DAC channel2 disabled
- 1: DAC channel2 enabled

*Note:* These bits are available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 15  Reserved, must be kept at reset value.

Bit 14  **CEN1**: DAC channel1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1 = 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled). Otherwise, the write operation is ignored.
- 0: DAC channel1 in Normal operating mode
- 1: DAC channel1 in calibration mode

Bit 13  **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.
- 0: DAC channel1 DMA Underrun Interrupt disabled
- 1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12  **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.
- 0: DAC channel1 DMA mode disabled
- 1: DAC channel1 DMA mode enabled

Bits 11:8  **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.
- 0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
- 0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
- 0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
- 0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
- 0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
- 0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
- 0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
- 0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
- 1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
- 1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
- 1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
- 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6  **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.
- 00: wave generation disabled
- 01: Noise wave generation enabled
- 1x: Triangle wave generation enabled

Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).
Bits 5:2 **TSEL1[3:0]:** DAC channel1 trigger selection
These bits select the external event used to trigger DAC channel1
0000: SWTRIG1
0001: dac_ch1_trg1
0010: dac_ch1_trg2
...
1111: dac_ch1_trg15
Refer to the trigger selection tables in Section 28.4.2: DAC pins and internal signals for details on trigger configuration and mapping.

*Note:* Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 1 **TEN1:** DAC channel1 trigger enable
This bit is set and cleared by software to enable/disable DAC channel1 trigger.
0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_hclk clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_hclk clock cycles later to the DAC_DOR1 register

*Note:* When software trigger is selected, the transfer from the DAC_DHR1 register to the DAC_DOR1 register takes only one dac_hclk clock cycle.

Bit 0 **EN1:** DAC channel1 enable
This bit is set and cleared by software to enable/disable DAC channel1.
0: DAC channel1 disabled
1: DAC channel1 enabled

### 28.7.2 DAC software trigger register (DAC_SWTRGR)

Address offset: 0x04
Reset value: 0x0000 0000
Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SWTRIG2**: DAC channel2 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger
1: Trigger

*Note*: This bit is cleared by hardware (one dac_hclk clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.

This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger
1: Trigger

*Note*: This bit is cleared by hardware (one dac_hclk clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

### 28.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

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

*DAC1DHRB[11:0]*

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

*DAC1DHR[11:0]*

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DAC1DHRB[11:0]**: DAC channel1 12-bit right-aligned data B

These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in Double data mode.

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

Bits 11:0 **DAC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel1.
28.7.4 DAC channel1 12-bit left aligned data holding register
(DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bits 31:20 DAC1DHRB[11:0]: DAC channel1 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in Double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 DAC1DHR[11:0]: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

28.7.5 DAC channel1 8-bit right aligned data holding register
(DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 DAC1DHRB[7:0]: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1 when the DAC operates in Double data mode.

Bits 7:0 DAC1DHR[7:0]: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.
28.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

This register is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Address offset: 0x14
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28</td>
<td>rw rw rw rw</td>
<td>DACC2DHRB[11:0]</td>
</tr>
<tr>
<td>27 26 25 24</td>
<td>rw rw rw rw</td>
<td></td>
</tr>
<tr>
<td>23 22 21 20</td>
<td>rw rw rw rw</td>
<td></td>
</tr>
<tr>
<td>19 18 17 16</td>
<td>rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DHRB[11:0]:** DAC channel2 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in DMA Double data mode.

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

Bits 11:0 **DACC2DHR[11:0]:** DAC channel2 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel2.

28.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

This register is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Address offset: 0x18
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28</td>
<td>rw rw rw rw</td>
<td>DACC2DHRB[11:0]</td>
</tr>
<tr>
<td>27 26 25 24</td>
<td>rw rw rw rw</td>
<td></td>
</tr>
<tr>
<td>23 22 21 20</td>
<td>rw rw rw rw</td>
<td></td>
</tr>
<tr>
<td>19 18 17 16</td>
<td>rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:20 **DACC2DHRB[11:0]:** DAC channel2 12-bit left-aligned data B

These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in Double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC2DHR[11:0]:** DAC channel2 12-bit left-aligned data

These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.
28.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

This register is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Address offset: 0x1C
Reset value: 0x0000 0000

Bits 31:16  Reserved, must be kept at reset value.
Bits 15:8  DAC2DHRB[7:0]: DAC channel2 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel2 when the DAC operates in Double data mode.

Bits 7:0  DAC2DHR[7:0]: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

28.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20
Reset value: 0x0000 0000

Bits 31:28  Reserved, must be kept at reset value.
Bits 27:16  DAC2DHR[11:0]: DAC channel2 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 15:12  Reserved, must be kept at reset value.
Bits 11:0  DAC1DHR[11:0]: DAC channel1 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.
28.7.10  Dual DAC 12-bit left aligned data holding register
(DAC_DHR12LD)

Address offset: 0x24
Reset value: 0x0000 0000

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

Bits 31:20  **DAC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 19:16  Reserved, must be kept at reset value.

Bits 15:4  **DAC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0  Reserved, must be kept at reset value.

28.7.11  Dual DAC 8-bit right aligned data holding register
(DAC_DHR8RD)

Address offset: 0x28
Reset value: 0x0000 0000

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

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:8  **DAC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0  **DAC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.
28.7.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C
Reset value: 0x0000 0000

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 DACC1DORB[11:0]: DAC channel1 data output
These bits are read-only. They contain data output for DAC channel1 B.

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

Bits 11:0 DACC1DOR[11:0]: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.

28.7.13 DAC channel2 data output register (DAC_DOR2)

This register is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Address offset: 0x30
Reset value: 0x0000 0000

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 DACC2DORB[11:0]: DAC channel2 data output
These bits are read-only. They contain data output for DAC channel2 B.

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

Bits 11:0 DACC2DOR[11:0]: DAC channel2 data output
These bits are read-only, they contain data output for DAC channel2.
28.7.14  DAC status register (DAC_SR)

Address offset: 0x34
Reset value: 0x0000 0000

Bit 31  BWST2: DAC channel2 busy writing sample time flag
This bit is systematically set just after Sample and hold mode enable. It is set each time the software writes the register DAC_SHSR2. It is cleared by hardware when the write operation of DAC_SHSR2 is complete. (It takes about 3 LSI/LSE periods of synchronization).
0: There is no write operation of DAC_SHSR2 ongoing: DAC_SHSR2 can be written
1: There is a write operation of DAC_SHSR2 ongoing: DAC_SHSR2 cannot be written

Notes: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 30  CAL_FLAG2: DAC channel2 calibration offset status
This bit is set and cleared by hardware.
0: Calibration trimming value is lower than the offset correction value
1: Calibration trimming value is equal or greater than the offset correction value

Notes: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 29  DMAUDR2: DAC channel2 DMA underrun flag
This bit is set by hardware and cleared by software (by writing it to 1).
0: No DMA underrun error condition occurred for DAC channel2
1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate).

Notes: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 28  DORSTAT2: DAC channel2 output register status bit
This bit is set and cleared by hardware. It is applicable only when the DAC operates in Double data mode.
0: DOR[11:0] is used actual DAC output
1: DORB[11:0] is used actual DAC output

Notes: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 27  DAC2RDY: DAC channel2 ready status bit
This bit is set and cleared by hardware.
0: DAC channel2 is not yet ready to accept the trigger nor output data
1: DAC channel2 is ready to accept the trigger or output data

Notes: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.
**28.7.15 DAC calibration control register (DAC_CCR)**

Address offset: 0x38

Reset value: 0x00XX 00XX

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

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

<table>
<thead>
<tr>
<th>OTRM2[4:0]</th>
<th>OTRM1[4:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bits 15** `BWST1`: DAC channel1 busy writing sample time flag

This bit is systematically set just after Sample and hold mode enable and is set each time the software writes the register DAC_SHSR1. It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI/LSE periods of synchronization).

- 0: There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written
- 1: There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

**Bit 14** `CAL_FLAG1`: DAC channel1 calibration offset status

This bit is set and cleared by hardware.

- 0: Calibration trimming value is lower than the offset correction value
- 1: Calibration trimming value is equal or greater than the offset correction value

**Bit 13** `DMAUDR1`: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

- 0: No DMA underrun error condition occurred for DAC channel1
- 1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

**Bit 12** `DORSTAT1`: DAC channel1 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in Double data mode.

- 0: DOR[11:0] is used actual DAC output
- 1: DORB[11:0] is used actual DAC output

**Bit 11** `DAC1RDY`: DAC channel1 ready status bit

This bit is set and cleared by hardware.

- 0: DAC channel1 is not yet ready to accept the trigger nor output data
- 1: DAC channel1 is ready to accept the trigger or output data

**Bits 10:0** Reserved, must be kept at reset value.

**Bits 31:21** Reserved, must be kept at reset value.

**Bits 20:16** `OTRIM2[4:0]`: DAC channel2 offset trimming value

These bits are available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

**Bits 15:5** Reserved, must be kept at reset value.

**Bits 4:0** `OTRIM1[4:0]`: DAC channel1 offset trimming value
28.7.16 DAC mode control register (DAC_MCR)

Address offset: 0x3C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:26</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 25</td>
<td>SINFOORMAT2: Enable signed format for DAC channel2</td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: Input data is in unsigned format</td>
</tr>
<tr>
<td></td>
<td>1: Input data is in signed format (2’s complement). The MSB bit represents the sign.</td>
</tr>
<tr>
<td></td>
<td>Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.</td>
</tr>
<tr>
<td>Bit 24</td>
<td>DMADouble2: DAC channel2 DMA double data mode</td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: DMA Normal mode selected</td>
</tr>
<tr>
<td></td>
<td>1: DMA Double data mode selected</td>
</tr>
<tr>
<td></td>
<td>Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.</td>
</tr>
</tbody>
</table>

Bits 23:19 Reserved, must be kept at reset value.

Bits 18:16 MODE2[2:0]: DAC channel2 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2 = 0 and bit CEN2 = 0 in the DAC_CR register). If EN2 = 1 or CEN2 = 1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel2 mode:

- DAC channel2 in Normal mode
  - 000: DAC channel2 is connected to external pin with Buffer enabled
  - 001: Reserved
  - 010: DAC channel2 is connected to external pin with buffer disabled
  - 011: Reserved
- DAC channel2 in Sample and hold mode
  - 100: DAC channel2 is connected to external pin with Buffer enabled
  - 101: Reserved
  - 110: DAC channel2 is connected to external pin with Buffer disabled
  - 111: Reserved

Note: This register can be modified only when EN2 = 0. Refer to Section 28.3: DAC implementation for the availability of DAC channel2.
Bits 15:14 **HFSEL[1:0]**: High frequency interface mode selection
- 00: High frequency interface mode disabled
- 01: High frequency interface mode enabled for AHB clock frequency > 80 MHz
- 10: High frequency interface mode enabled for AHB clock frequency > 160 MHz
- 11: Reserved

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **SINFORMAT1**: Enable signed format for DAC channel1
This bit is set and cleared by software.
0: Input data is in unsigned format
1: Input data is in signed format (2’s complement). The MSB bit represents the sign.

Bit 8 **DMADouble1**: DAC channel1 DMA double data mode
This bit is set and cleared by software.
0: DMA Normal mode selected
1: DMA Double data mode selected

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

Bits 2:0 **MODE[2:0]**: DAC channel1 mode
These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.
They can be set and cleared by software to select the DAC channel1 mode:
- DAC channel1 in Normal mode
  000: DAC channel1 is connected to external pin with Buffer enabled
  001: Reserved
  010: DAC channel1 is connected to external pin with Buffer disabled
  011: Reserved
- DAC channel1 in sample & hold mode
  100: DAC channel1 is connected to external pin with Buffer enabled
  101: Reserved
  110: DAC channel1 is connected to external pin with Buffer disabled
  111: Reserved

Note: This register can be modified only when EN1 = 0.

### 28.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1)

Address offset: 0x40
Reset value: 0x0000 0000

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

**TSAMPLE1[9:0]**

<table>
<thead>
<tr>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
28.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2)

This register is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Address offset: 0x44
Reset value: 0x0000 0000

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

Bits 9:0 **TSAMPLE1[9:0]**: DAC channel1 sample time (only valid in Sample and hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. in the latter case, the write can be done only when BWST1 of DAC_SR register is low, if BWST1 = 1, the write operation is ignored.

Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (TSAMPLE1[9:0] + 1) x LSI/LSE clock period.

28.7.19 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48
Reset value: 0x0001 0001

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

Bits 9:0 **TSAMPLE2[9:0]**: DAC channel2 sample time (only valid in Sample and hold mode)

These bits can be written when the DAC channel2 is disabled or also during normal operation. in the latter case, the write can be done only when BWST2 of DAC_SR register is low, if BWST2 = 1, the write operation is ignored.

Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (TSAMPLE1[9:0] + 1) x LSI/LSE clock period.
Bits 31:26  Reserved, must be kept at reset value.

Bits 25:16  **THOLD2[9:0]**: DAC channel2 hold time (only valid in Sample and hold mode).
            Hold time = (THOLD[9:0]) x LSI/LSE clock period
            *Note:* This register can be modified only when EN2 = 0.
            *Note:* These bits are available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bits 15:10  Reserved, must be kept at reset value.

Bits 9:0   **THOLD1[9:0]**: DAC channel1 hold time (only valid in Sample and hold mode)
            Hold time = (THOLD[9:0]) x LSI/LSE clock period
            *Note:* This register can be modified only when EN1 = 0.

*Note:* These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

### 28.7.20 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C

Reset value: 0x0001 0001

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

Bits 31:24  Reserved, must be kept at reset value.

Bits 23:16  **TREFRESH2[7:0]**: DAC channel2 refresh time (only valid in Sample and hold mode)
            Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period
            *Note:* This register can be modified only when EN2 = 0.
            *Note:* These bits are available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bits 15:8  Reserved, must be kept at reset value.

Bits 7:0   **TREFRESH1[7:0]**: DAC channel1 refresh time (only valid in Sample and hold mode)
            Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period
            *Note:* This register can be modified only when EN1 = 0.

*Note:* These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.
# 28.7.21 DAC register map

Table 273 summarizes the DAC registers.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name reset value</th>
<th>Register name reset value</th>
<th>Offset</th>
<th>Register name reset value</th>
<th>Register name reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>0x08</td>
<td>DAC_DHR12R1</td>
<td>DACC1DHRB[11:0] DACC1DHR[11:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>0x0C</td>
<td>DAC_DHR12L1</td>
<td>DACC1DHRB[11:0] DACC1DHR[11:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>0x10</td>
<td>DAC_DHR8R1</td>
<td>DACC1DHRB[7:0] DACC1DHR[7:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>0x14</td>
<td>DAC_DHR12R2</td>
<td>DACC2DHRB[11:0] DACC2DHR[11:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>0x18</td>
<td>DAC_DHR12L2</td>
<td>DACC2DHRB[11:0] DACC2DHR[11:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>0x1C</td>
<td>DAC_DHR8R2</td>
<td>DACC2DHRB[7:0] DACC2DHR[7:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>0x20</td>
<td>DAC_DHR12RD</td>
<td>DACC2DHR[11:0] DACC1DHR[11:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>0x24</td>
<td>DAC_DHR12LD</td>
<td>DACC2DHR[11:0] DACC1DHR[11:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>0x28</td>
<td>DAC_DHR8RD</td>
<td>DACC2DHR[7:0] DACC1DHR[7:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>0x2C</td>
<td>DAC_DOR1</td>
<td>DACC1DORB[11:0] DACC1DOR[11:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>Reset value 0x00000000000000000000000000000000</td>
<td>0x30</td>
<td>DAC_DOR2</td>
<td>DACC2DORB[11:0] DACC2DOR[11:0]</td>
</tr>
</tbody>
</table>

Table 273. DAC register map and reset values
Refer to *Section 2.3 on page 111* for the register boundary addresses.
29 Voltage reference buffer (VREFBUF)

29.1 Introduction

The devices embed a voltage reference buffer which can be used as voltage reference for ADCs and also as voltage reference for external components through the VREF+ pin.

29.2 VREFBUF implementation

The table below describes the VREFBUF voltages typical values:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VREFBUF0</td>
<td>2.5 V</td>
</tr>
<tr>
<td>VREFBUF1</td>
<td>2.048 V</td>
</tr>
<tr>
<td>VREFBUF2</td>
<td>1.8 V</td>
</tr>
</tbody>
</table>

Note: Refer to the product datasheet for more details.

29.3 VREFBUF functional description

The internal voltage reference buffer is an operational amplifier, with programmable gain. The amplifier input is connected to the internal voltage reference \( V_{\text{REFINT}} \). The VREFBUF supports four voltages\(^8\), which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 000: VREFBUF0 voltage selected.
- VRS = 001: VREFBUF1 voltage selected.
- VRS = 010: VREFBUF2 voltage selected.

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:
After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

### 29.4 VREFBUF trimming

The VREFBUF output voltage is factory-calibrated by ST. At reset, and each time the VRS setting is changed, the calibration data is automatically loaded to the TRIM register.

Optionally user can trim the output voltage by changing the TRIM register bits directly. In this case, the VRS setting has no more effect on the TRIM register until the device is reset.

### 29.5 VREFBUF registers

#### 29.5.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x00
Reset value: 0x0000 0002

<table>
<thead>
<tr>
<th>ENVR</th>
<th>HIZ</th>
<th>VREF buffer configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>VREFBUF buffer off mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– V\textsubscript{REF}+ pin pulled-down to V\textsubscript{SSA}</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>External voltage reference mode (default value):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREFBUF buffer off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– V\textsubscript{REF}+ pin input mode</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Internal voltage reference mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREFBUF buffer on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– V\textsubscript{REF}+ pin connected to VREFBUF buffer output</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Hold mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VREF is enable without output buffer, VREF+ pin voltage is hold with the external capacitor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– VRR detection disabled and VRR bit keeps last state</td>
</tr>
</tbody>
</table>

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

### 29.4 VREFBUF trimming

The VREFBUF output voltage is factory-calibrated by ST. At reset, and each time the VRS setting is changed, the calibration data is automatically loaded to the TRIM register.

Optionally user can trim the output voltage by changing the TRIM register bits directly. In this case, the VRS setting has no more effect on the TRIM register until the device is reset.

#### 29.5 VREFBUF registers

#### 29.5.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x00
Reset value: 0x0000 0002

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
a. The minimum \( V\text{\textsubscript{DDA}} \) voltage depends on VRS setting, refer to the product datasheet.
29.5.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 VRS[2:0]: Voltage reference scale

These bits select the value generated by the voltage reference buffer.

VRS = 000: VREFBUF0 voltage selected.
VRS = 001: VREFBUF1 voltage selected.
VRS = 010: VREFBUF2 voltage selected.
Others: Reserved

Note: Refer to the product datasheet for each VREFBUFx voltage setting value.
The software can program this bitfield only when the VREFBUF is disabled (ENVR=0).

Bit 3 VRR: Voltage reference buffer ready

0: the voltage reference buffer output is not ready.
1: the voltage reference buffer output reached the requested level.

Bit 2 Reserved, must be kept at reset value.

Bit 1 HIZ: High impedance mode

This bit controls the analog switch to connect or not the VREF+ pin.
0: VREF+ pin is internally connected to the voltage reference buffer output.
1: VREF+ pin is high impedance.

Refer to Table 275: VREF buffer modes for the mode descriptions depending on ENVR bit configuration.

Bit 0 ENVR: Voltage reference buffer mode enable

This bit is used to enable the voltage reference buffer mode.
0: Internal voltage reference mode disable (external voltage reference mode).
1: Internal voltage reference mode (reference buffer enable or hold mode) enable.
Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 TRIM[5:0]: Trimming code

The TRIM code is a 6-bit unsigned data (minimum 000000, maximum 111111) that is set and updated according the mechanism described below.

- **Reset:**
  TRIM[5:0] is automatically initialized with the VRS = 0 trimming value stored in the flash memory during the production test.

- **VRS change:**
  TRIM[5:0] is automatically initialized with the trimming value (corresponding to VRS setting) stored in the flash memory during the production test.

- **Write in TRIM[5:0]:**
  User can modify the TRIM[5:0] with an arbitrary value. This is permanently disabling the control of the trimming value with VRS (until the device is reset).

*Note:* If the user application performs the trimming, the trimming code must start from 000000 to 111111 in ascending order.

### 29.5.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 0x00   | VREFBUF_CSR   | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             |
|        | Reset value   | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 1             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             | 0             |
| 0x04   | VREFBUF_CCR   | TRIM[5:0]     | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             |
|        | Reset value   | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             | x             |

Refer to Section 2.3 for the register boundary addresses.
30 Digital camera interface (DCMI)

30.1 Introduction

The digital camera is a synchronous parallel interface able to receive a high-speed data flow from an external 8-, 10-, 12- or 14-bit CMOS camera module. It supports different data formats: YCbCr4:2:2/RGB565 progressive video and compressed data (JPEG).

30.2 DCMI main features

- 8-, 10-, 12- or 14-bit parallel interface
- Embedded/external line and frame synchronization
- Continuous or snapshot mode
- Crop feature
- Supports the following data formats:
  - 8/10/12/14-bit progressive video: either monochrome or raw Bayer
  - YCbCr 4:2:2 progressive video
  - RGB 565 progressive video
  - Compressed data: JPEG

30.3 DCMI functional description

The digital camera interface is a synchronous parallel interface that can receive high-speed data flows. It consists of up to 14 data lines (DCMI_D[13:0]) and a pixel clock line (DCMI_PIXCLK). The pixel clock has a programmable polarity, so that data can be captured on either the rising or the falling edge of the pixel clock.

The data are packed into a 32-bit data register (DCMI_DR) and then transferred through a general-purpose DMA channel. The image buffer is managed by the DMA, not by the camera interface.

The data received from the camera can be organized in lines/frames (raw YUB/RGB/Bayer modes) or can be a sequence of JPEG images. To enable JPEG image reception, the JPEG bit (bit 3 of DCMI_CR register) must be set.

The data flow is synchronized either by hardware using the optional DCMI_HSYNC (horizontal synchronization) and DCMI_VSYNC (vertical synchronization) signals or by synchronization codes embedded in the data flow.
30.3.1 DCMI block diagram

*Figure 273* shows the DCMI block diagram.

![Figure 273. DCMI block diagram](image)

30.3.2 DCMI pins and internal signals

The following table shows DCMI pins.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>DCMI_D[7:0]</td>
<td>Inputs</td>
<td>DCMI data</td>
</tr>
<tr>
<td>10 bits</td>
<td>DCMI_D[9:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 bits</td>
<td>DCMI_D[11:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 bits</td>
<td>DCMI_D[13:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DCMI_PIXCLK</td>
<td>Input</td>
<td>Pixel clock</td>
</tr>
<tr>
<td></td>
<td>DCMI_HSYNC</td>
<td>Input</td>
<td>Horizontal synchronization / Data valid</td>
</tr>
<tr>
<td></td>
<td>DCMI_VSYNC</td>
<td>Input</td>
<td>Vertical synchronization</td>
</tr>
</tbody>
</table>

The following table shows DCMI internal signals.

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dcmi_dma</td>
<td>Output</td>
<td>DCMI DMA request</td>
</tr>
<tr>
<td>dcmi_it</td>
<td>Output</td>
<td>DCMI interrupt request</td>
</tr>
<tr>
<td>dcmi_hclk</td>
<td>Input</td>
<td>DCMI interface clock</td>
</tr>
</tbody>
</table>
30.3.3 DCMI clocks

The digital camera interface uses two clock domains, DCMI_PIXCLK and HCLK. The signals generated with DCMI_PIXCLK are sampled on the rising edge of HCLK once they are stable. An enable signal is generated in the HCLK domain, to indicate that data coming from the camera are stable and can be sampled. The maximum DCMI_PIXCLK period must be higher than 2.5 HCLK periods.

30.3.4 DCMI DMA interface

The DMA interface is active when the CAPTURE bit of the DCMI_CR register is set. A DMA request is generated each time the camera interface receives a complete 32-bit data block in its register.

30.3.5 DCMI physical interface

The interface is composed of 11/13/15/17 inputs. Only the Slave mode is supported.

The camera interface can capture 8-bit, 10-bit, 12-bit or 14-bit data depending on the EDM[1:0] bits of the DCMI_CR register. If less than 14 bits are used, the unused input pins must be connected to ground.

DCMI pins are shown in Table 277.

The data are synchronous with DCMI_PIXCLK and change on the rising/falling edge of the pixel clock depending on the polarity.

The DCMI_HSYNC signal indicates the start/end of a line.

The DCMI_VSYNC signal indicates the start/end of a frame.

**Figure 274. DCMI signal waveforms**

![DCMI signal waveforms](ai15606c)

1. The capture edge of DCMI_PIXCLK is the falling edge, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.

2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

8-bit data

When EDM[1:0] = 00 in DCMI_CR the interface captures 8 LSBs at its input (DCMI_D[7:0]) and stores them as 8-bit data. The DCMI_D[13:8] inputs are ignored. In this case, to capture a 32-bit word, the camera interface takes four pixel clock cycles.

The first captured data byte is placed in the LSB position in the 32-bit word and the 4th captured data byte is placed in the MSB position in the 32-bit word. The table below gives an example of the positioning of captured data bytes in two 32-bit words.
10-bit data

When EDM[1:0] = 01 in DCMI_CR, the camera interface captures 10-bit data at its input DCMI_D[9:0] and stores them as the 10 least significant bits of a 16-bit word. The remaining most significant bits of the DCMI_DR register (bits 11 to 15) are cleared to zero. So, in this case, a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2\textsuperscript{nd} captured data are placed in the MSB position in the 32-bit word as shown in the table below.

| Table 279. Positioning of captured data bytes in 32-bit words (8-bit width) |
|------------------|------------------|------------------|------------------|------------------|
| Byte address    | 31:24            | 23:16            | 15:8             | 7:0              |
| 0               | D_{n+3}[7:0]     | D_{n+2}[7:0]     | D_{n+1}[7:0]     | D_{n}[7:0]       |
| 4               | D_{n+7}[7:0]     | D_{n+6}[7:0]     | D_{n+5}[7:0]     | D_{n+4}[7:0]     |

12-bit data

When EDM[1:0] = 10 in DCMI_CR, the camera interface captures the 12-bit data at its input DCMI_D[11:0] and stores them as the 12 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2\textsuperscript{nd} captured data are placed in the MSB position in the 32-bit word as shown in the table below.

| Table 280. Positioning of captured data bytes in 32-bit words (10-bit width) |
|------------------|------------------|------------------|------------------|------------------|
| Byte address    | 31:26            | 25:16            | 15:10            | 9:0              |
| 0               | 0                | D_{n+1}[9:0]     | 0                | D_{n}[9:0]       |
| 4               | 0                | D_{n+3}[9:0]     | 0                | D_{n+2}[9:0]     |

14-bit data

When EDM[1:0] = 11 in DCMI_CR, the camera interface captures the 14-bit data at its input DCMI_D[13:0] and stores them as the 14 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2\textsuperscript{nd} captured data are placed in the MSB position in the 32-bit word as shown in the table below.

| Table 281. Positioning of captured data bytes in 32-bit words (12-bit width) |
|------------------|------------------|------------------|------------------|------------------|
| Byte address    | 31:28            | 27:16            | 15:12            | 11:0             |
| 0               | 0                | D_{n+1}[11:0]    | 0                | D_{n}[11:0]      |
| 4               | 0                | D_{n+3}[11:0]    | 0                | D_{n+2}[11:0]    |
30.3.6 DCMI synchronization

The digital camera interface supports embedded or hardware (DCMI_HSYNC and DCMI_VSYNC) synchronization. When embedded synchronization is used, it is up to the digital camera module to make sure that the 0x00 and 0xFF values are used ONLY for synchronization (not in data). Embedded synchronization codes are supported only for the 8-bit parallel data interface width (that is, in the DCMI_CR register, the EDM[1:0] bits must be cleared).

For compressed data, the DCMI supports only the hardware synchronization mode. In this case, DCMI_VSYNC is used as a start/end of the image, and DCMI_HSYNC is used as a Data Valid signal. Figure 275 shows the corresponding timing diagram.

Table 282. Positioning of captured data bytes in 32-bit words (14-bit width)

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:30</th>
<th>29:16</th>
<th>15:14</th>
<th>13:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>D_{p+1}[13:0]</td>
<td>0</td>
<td>D_{p}[13:0]</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>D_{p+3}[13:0]</td>
<td>0</td>
<td>D_{p+2}[13:0]</td>
</tr>
</tbody>
</table>

Figure 275. Timing diagram

Hardware synchronization mode

In hardware synchronization mode, the two synchronization signals (DCMI_HSYNC/DCMI_VSYNC) are used.

Depending on the camera module/mode, data may be transmitted during horizontal/vertical synchronization periods. The DCMI_HSYNC/DCMI_VSYNC signals act like blanking signals since all the data received during DCMI_HSYNC/DCMI_VSYNC active periods are ignored.

In order to correctly transfer images into the DMA/RAM buffer, data transfer is synchronized with the DCMI_VSYNC signal. When the hardware synchronization mode is selected, and...
capture is enabled (CAPTURE bit set in DCMI_CR), data transfer is synchronized with the deactivation of the DCMI_VSYNC signal (next start of frame).

Transfer can then be continuous, with successive frames transferred by DMA to successive buffers or the same/circular buffer. To allow the DMA management of successive frames, a VSIF (Vertical synchronization interrupt flag) is activated at the end of each frame.

**Embedded data synchronization mode**

In this synchronization mode, the data flow is synchronized using 32-bit codes embedded in the data flow. These codes use the 0x00/0xFF values that are not used in data anymore. There are 4 types of codes, all with a 0xFF0000XY format. The embedded synchronization codes are supported only in 8-bit parallel data width capture (in the DCMI_CR register, the EDM[1:0] bits must be cleared). For other data widths, this mode generates unpredictable results and must not be used.

*Note: Camera modules can have 8 such codes (in interleaved mode). For this reason, the interleaved mode is not supported by the camera interface (otherwise, every other half-frame would be discarded).*

- **Mode 2**
  - Four embedded codes signal the following events
    - Frame start (FS)
    - Frame end (FE)
    - Line start (LS)
    - Line end (LE)

  The XY values in the 0xFF0000XY format of the four codes are programmable (see Section 30.5.7: DCMI embedded synchronization code register (DCMI_ESCR)).

  A 0xFF value programmed as a "frame end" means that all the unused codes are interpreted as valid frame end codes.

  In this mode, once the camera interface has been enabled, the frame capture starts after the first occurrence of the frame end (FE) code followed by a frame start (FS) code.

- **Mode 1**
  - An alternative coding is the camera mode 1. This mode is ITU656 compatible.

    The codes signal another set of events:
    - SAV (active line) - line start
    - EAV (active line) - line end
    - SAV (blanking) - end of line during interframe blanking period
    - EAV (blanking) - end of line during interframe blanking period

    This mode can be supported by programming the following codes:
    - FS ≤ 0xFF
    - FE ≤ 0xFF
    - LS ≤ SAV (active)
    - LE ≤ EAV (active)

    An embedded unmask code is also implemented for frame/line start and frame/line end codes. Using it, it is possible to compare only the selected unmasked bits with the programmed code. A bit can therefore be selected to compare in the embedded code and
detect a frame/line start or frame/line end. This means that there can be different codes for the frame/line start and frame/line end with the unmasked bit position remaining the same.

**Example**

FS = 0xA5

Unmask code for FS = 0x10

In this case the frame start code is embedded in the bit 4 of the frame start code.

### 30.3.7 DCMI capture modes

This interface supports two types of capture: snapshot (single frame) and continuous grab.

**Snapshot mode (single frame)**

In this mode, a single frame is captured (CM = 1 of the DCMI_CR register). After the CAPTURE bit is set in DCMI_CR, the interface waits for the detection of a start of frame before sampling the data. The camera interface is automatically disabled (CAPTURE bit cleared in DCMI_CR) after receiving the first complete frame. An interrupt is generated (IT_FRAME) if it is enabled.

In case of an overrun, the frame is lost and the CAPTURE bit is cleared.

![Figure 276. Frame capture waveforms in snapshot mode](image)

1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

**Continuous grab mode**

In this mode (CM bit = 0 in DCMI_CR), once the CAPTURE bit has been set in DCMI_CR, the grabbing process starts on the next DCMI_VSYNC or embedded frame start depending on the mode. The process continues until the CAPTURE bit is cleared in DCMI_CR. Once the CAPTURE bit has been cleared, the grabbing process continues until the end of the current frame.
1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.

2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

In continuous grab mode, the FCRC[1:0] bits in DCMI_CR can be configured to grab all pictures, every second picture or one out of four pictures to decrease the frame capture rate.

**Note:** *In the hardware synchronization mode (ESS = 0 in DCMI_CR), the IT_VSYNC interrupt is generated (if enabled) even when CAPTURE = 0 in DCMI_CR so, to reduce the frame capture rate even further, the IT_VSYNC interrupt can be used to count the number of frames between 2 captures in conjunction with the Snapshot mode. This is not allowed by embedded data synchronization mode.*

### 30.3.8 DCMI crop feature

With the crop feature, the camera interface can select a rectangular window from the received image. The start (upper left corner) coordinates and size (horizontal dimension in number of pixel clocks and vertical dimension in number of lines) are specified using two 32-bit registers (DCMI_CWSTRT and DCMI_CWSIZE). The size of the window is specified in number of pixel clocks (horizontal dimension) and in number of lines (vertical dimension).

**Figure 278. Coordinates and size of the window after cropping**

These registers specify the coordinates of the starting point of the capture window as a line number (in the frame, starting from 0) and a number of pixel clocks (on the line, starting from 0), and the size of the window as a line number and a number of pixel clocks. The CAPCNT value can only be a multiple of 4 (two least significant bits are forced to 0) to allow the correct transfer of data through the DMA.
If the DCMI_VSYNC signal goes active before the number of lines is specified in the DCMI_CWSIZE register, then the capture stops and an IT_FRAME interrupt is generated when enabled.

**Figure 279. Data capture waveforms**

1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

### 30.3.9 DCMI JPEG format

To allow JPEG image reception, it is necessary to set the JPEG bit of the DCMI_CR register. JPEG images are not stored as lines and frames, so the DCMI_VSYNC signal is used to start the capture while DCMI_HSYNC serves as a data enable signal. The number of bytes in a line may not be a multiple of 4. This case must be carefully handled since a DMA request is generated each time a complete 32-bit word has been constructed from the captured data. When an end of frame is detected and the 32-bit word to be transferred has not been completely received, the remaining data are padded with zeros and a DMA request is generated.

The crop feature and embedded synchronization codes cannot be used in JPEG format.

### 30.3.10 DCMI FIFO

A 8-word FIFO is implemented to manage data rate transfers on the AHB. The DCMI features a simple FIFO controller with a read pointer incremented each time the camera interface reads from the AHB, and a write pointer incremented each time the camera interface writes to the FIFO. There is no overrun protection to prevent the data from being overwritten if the AHB interface does not sustain the data transfer rate.

In case of overrun or errors in the synchronization signals, the FIFO is reset and the DCMI interface waits for a new start of frame.
30.3.11 DCMI data format description

Data formats

Three types of data are supported:

- 8/10/12/14-bit progressive video: either monochrome or raw Bayer format
- YCbCr 4:2:2 progressive video
- RGB565 progressive video. A pixel coded in 16 bits (5 bits for blue, 5 bits for red, 6 bits for green) takes two clock cycles to be transferred.

Compressed data: JPEG

For B&W (black and white), YCbCr or RGB data, the maximum input size is 2048 × 2048 pixels. No limit in JPEG compressed mode.

For monochrome, RGB and YCbCr, the frame buffer is stored in raster mode. 32-bit words are used. Only the little-endian format is supported.

![Figure 280. Pixel raster scan order](image)

Monochrome format

Characteristics:

- Raster format
- 8 bits per pixel

The table below shows how the data are stored.

| Table 283. Data storage in monochrome progressive video format |
|---|---|---|---|---|
| Byte address | 31:24 | 23:16 | 15:8 | 7:0 |
| 0 | n + 3 | n + 2 | n + 1 | n |
| 4 | n + 7 | n + 6 | n + 5 | n + 4 |

RGB format

Characteristics:

- Raster format
- RGB
- Interleaved: one buffer: R, G and B interleaved (such as BRGBRGBRG)
- Optimized for display output
The RGB planar format is compatible with standard OS frame buffer display formats. Only 16 BPP (bits per pixel): RGB565 (2 pixels per 32-bit word) is supported.

The 24 BPP (palletized format) and gray-scale formats are not supported. Pixels are stored in a raster scan order, that is from top to bottom for pixel rows, and from left to right within a pixel row. Pixel components are R (red), G (green) and B (blue). All components have the same spatial resolution (4:4:4 format). A frame is stored in a single part, with the components interleaved on a pixel basis.

The table below shows how the data are stored.

**Table 284. Data storage in RGB progressive video format**

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:27</th>
<th>26:21</th>
<th>20:16</th>
<th>15:11</th>
<th>10:5</th>
<th>4:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Red n + 1</td>
<td>Green n + 1</td>
<td>Blue n + 1</td>
<td>Red n</td>
<td>Green n</td>
<td>Blue n</td>
</tr>
<tr>
<td>4</td>
<td>Red n + 4</td>
<td>Green n + 3</td>
<td>Blue n + 3</td>
<td>Red n + 2</td>
<td>Green n + 2</td>
<td>Blue n + 2</td>
</tr>
</tbody>
</table>

**YCbCr format**

Characteristics:
- Raster format
- YCbCr 4:2:2
- Interleaved: one buffer: Y, Cb and Cr interleaved (such as CbYCrYCbYCr)

Pixel components are Y (luminance or "luma"), Cb and Cr (chrominance or "chroma" blue and red). Each component is encoded in 8 bits. Luma and chroma are stored together (interleaved) as shown in the table below.

**Table 285. Data storage in YCbCr progressive video format**

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:24</th>
<th>23:16</th>
<th>15:8</th>
<th>7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Y n + 1</td>
<td>Cr n</td>
<td>Y n</td>
<td>Cb n</td>
</tr>
<tr>
<td>4</td>
<td>Y n + 3</td>
<td>Cr n + 2</td>
<td>Y n + 2</td>
<td>Cb n + 2</td>
</tr>
</tbody>
</table>

**YCbCr format - Y only**

Characteristics:
- Raster format
- YCbCr 4:2:2
- The buffer only contains Y information - monochrome image

Pixel components are Y (luminance or "luma"), Cb and Cr (chrominance or "chroma" blue and red). In this mode, the chroma information is dropped. Only the luma component of each pixel, encoded in 8 bits, is stored as shown in Table 286.

The result is a monochrome image having the same resolution as the original YCbCr data.
Half resolution image extraction

This is a modification of the previous reception modes, being applicable to monochrome, RGB or Y extraction modes.

This mode is used to only store a half resolution image. It is selected through OELS and LSM control bits.

### 30.4 DCMI interrupts

Five interrupts are generated. All interrupts are maskable by software. The global interrupt (dcmi_it) is the OR of all the individual interrupts. The table below gives the list of all interrupts.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exits Sleep mode</th>
<th>Exists Stop and Standby modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>dcmi_it</td>
<td>End of line</td>
<td>LINE_RIS</td>
<td>LINE_IE</td>
<td>Set LINE_ISC</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>End of frame capture</td>
<td>FRAME_RIS</td>
<td>FRAME_IE</td>
<td>Set FRAME_ISC</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Overrun of data reception</td>
<td>OVR_RIS</td>
<td>OVR_IE</td>
<td>Set OVR_ISC</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Synchronization frame</td>
<td>VSYNC_RIS</td>
<td>VSYNC_IE</td>
<td>Set VSYNC_ISC</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Detection of an error in the embedded synchronization frame detection</td>
<td>ERR_RIS</td>
<td>ERR_IE</td>
<td>Set ERR_ISC</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
30.5 DCMI registers

Refer to Section 1.2 on page 101 for list of abbreviations used in register descriptions. All DCMI registers must be accessed as 32-bit words, otherwise a bus error occurs.

30.5.1 DCMI control register (DCMI_CR)

Address offset: 0x00
Reset value: 0x0000 0000

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

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 OELS: Odd/Even Line Select (Line Select Start)
This bit works in conjunction with the LSM field (LSM = 1).
0: Interface captures first line after the frame start, second one being dropped.
1: Interface captures second line from the frame start, first one being dropped.

Bit 19 LSM: Line Select mode
0: Interface captures all received lines.
1: Interface captures one line out of two.

Bit 18 OEBS: Odd/Even Byte Select (Byte Select Start)
This bit works in conjunction with BSM field (BSM ≠ 00).
0: Interface captures first data (byte or double byte) from the frame/line start, second one being dropped.
1: Interface captures second data (byte or double byte) from the frame/line start, first one being dropped.

Bits 17:16 BSM[1:0]: Byte Select mode
00: Interface captures all received data.
01: Interface captures every other byte from the received data.
10: Interface captures one byte out of four.
11: Interface captures two bytes out of four.

Note: This mode only works for EDM[1:0] = 00. For all other EDM values, this field must be programmed to the reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 ENABLE: DCMI enable
0: DCMI disabled
1: DCMI enabled

Note: The DCMI configuration registers must be programmed correctly before enabling this bit.

Bits 13:12 Reserved, must be kept at reset value.
Bits 11:10 **EDM[1:0]**: Extended data mode
00: Interface captures 8-bit data on every pixel clock.
01: Interface captures 10-bit data on every pixel clock.
10: Interface captures 12-bit data on every pixel clock.
11: Interface captures 14-bit data on every pixel clock.

Bits 9:8 **FCRC[1:0]**: Frame capture rate control
These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode. They are ignored in snapshot mode.
00: All frames are captured.
01: Every alternate frame captured (50% bandwidth reduction)
10: One frame out of four captured (75% bandwidth reduction)
11: reserved

Bit 7 **VSPOL**: Vertical synchronization polarity
This bit indicates the level on the DCMI_VSYNC pin when the data are not valid on the parallel interface.
0: DCMI_VSYNC active low
1: DCMI_VSYNC active high

Bit 6 **HSPOL**: Horizontal synchronization polarity
This bit indicates the level on the DCMI_HSYNC pin when the data are not valid on the parallel interface.
0: DCMI_HSYNC active low
1: DCMI_HSYNC active high

Bit 5 **PCKPOL**: Pixel clock polarity
This bit configures the capture edge of the pixel clock.
0: Falling edge active
1: Rising edge active

Bit 4 **ESS**: Embedded synchronization select
0: Hardware synchronization data capture (frame/line start/stop) is synchronized with the DCMI_HSYNC/DCMI_VSYNC signals.
1: Embedded synchronization data capture is synchronized with synchronization codes embedded in the data flow.
*Note*: Valid only for 8-bit parallel data. HSPOL/VSPOL are ignored when the ESS bit is set. This bit is disabled in JPEG mode.

Bit 3 **JPEG**: JPEG format
0: Uncompressed video format
1: This bit is used for JPEG data transfers. The DCMI_HSYNC signal is used as data enable. The crop and embedded synchronization features (ESS bit) cannot be used in this mode.

Bit 2 **CROP**: Crop feature
0: The full image is captured. In this case the total number of bytes in an image frame must be a multiple of four.
1: Only the data inside the window specified by the crop register is captured. If the size of the crop window exceeds the picture size, then only the picture size is captured.

Bit 1 **CM**: Capture mode
0: Continuous grab mode - The received data are transferred into the destination memory through the DMA. The buffer location and mode (linear or circular buffer) is controlled through the system DMA.
1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame and then transfers a single frame through the DMA. At the end of the frame, the CAPTURE bit is automatically reset.
Bit 0  **CAPTURE**: Capture enable
   0: Capture disabled
   1: Capture enabled
The camera interface waits for the first start of frame, then a DMA request is generated to transfer the received data into the destination memory.
In snapshot mode, the CAPTURE bit is automatically cleared at the end of the first frame received.
In continuous grab mode, if the software clears this bit while a capture is ongoing, the bit is effectively cleared after the frame end.
*Note*: The DMA controller and all DCMI configuration registers must be programmed correctly before enabling this bit.

### 30.5.2  DCMI status register (DCMI_SR)

Address offset: 0x04
Reset value: 0x0000 0000

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

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

Bit 2  **FNE**: FIFO not empty
   This bit gives the status of the FIFO.
   1: FIFO contains valid data.
   0: FIFO empty

Bit 1  **VSYNC**: Vertical synchronization
   This bit gives the state of the DCMI_VSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:
   0: active frame
   1: synchronization between frames
   In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

Bit 0  **HSYNC**: Horizontal synchronization
   This bit gives the state of the DCMI_HSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:
   0: active line
   1: synchronization between lines
   In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.
30.5.3 DCMI raw interrupt status register (DCMI_RIS)

DCMI_RIS gives the raw interrupt status and is accessible in read only. When read, this register returns the status of the corresponding interrupt before masking with the DCMI_IER register value.

Address offset: 0x08

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>4</td>
<td><strong>LINE_RIS</strong>: Line raw interrupt status</td>
</tr>
<tr>
<td></td>
<td>This bit gets set when the DCMI_HSYNC signal changes from the inactive state to the active state. It goes high even if the line is not valid.</td>
</tr>
<tr>
<td></td>
<td>In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMI_CR is set.</td>
</tr>
<tr>
<td></td>
<td>It is cleared by setting the LINE_ISC bit of the DCMI_ICR register.</td>
</tr>
<tr>
<td>3</td>
<td><strong>VSYNC_RIS</strong>: DCMI_VSYNC raw interrupt status</td>
</tr>
<tr>
<td></td>
<td>This bit is set when the DCMI_VSYNC signal changes from the inactive state to the active state.</td>
</tr>
<tr>
<td></td>
<td>In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMI_CR.</td>
</tr>
<tr>
<td></td>
<td>It is cleared by setting the VSYNC_ISC bit of the DCMI_ICR register.</td>
</tr>
<tr>
<td>2</td>
<td><strong>ERR_RIS</strong>: Synchronization error raw interrupt status</td>
</tr>
<tr>
<td></td>
<td>0: No synchronization error detected</td>
</tr>
<tr>
<td></td>
<td>1: Embedded synchronization characters are not received in the correct order.</td>
</tr>
<tr>
<td></td>
<td>This bit is valid only in the embedded synchronization mode. It is cleared by setting the ERR_ISC bit of the DCMI_ICR register.</td>
</tr>
<tr>
<td></td>
<td>Note: This bit is available only in embedded synchronization mode.</td>
</tr>
<tr>
<td>1</td>
<td><strong>OVR_RIS</strong>: Overrun raw interrupt status</td>
</tr>
<tr>
<td></td>
<td>0: No data buffer overrun occurred</td>
</tr>
<tr>
<td></td>
<td>1: A data buffer overrun occurred and the data FIFO is corrupted.</td>
</tr>
<tr>
<td></td>
<td>The bit is cleared by setting the OVR_ISC bit of the DCMI_ICR register.</td>
</tr>
<tr>
<td>0</td>
<td><strong>FRAME_RIS</strong>: Capture complete raw interrupt status</td>
</tr>
<tr>
<td></td>
<td>0: No new capture</td>
</tr>
<tr>
<td></td>
<td>1: A frame has been captured.</td>
</tr>
<tr>
<td></td>
<td>This bit is set when a frame or window has been captured.</td>
</tr>
<tr>
<td></td>
<td>In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (for example window cropped outside the frame).</td>
</tr>
<tr>
<td></td>
<td>The bit is cleared by setting the FRAME_ISC bit of the DCMI_ICR register.</td>
</tr>
</tbody>
</table>
30.5.4 **DCMI interrupt enable register (DCMI_IER)**

The DCMI_IER register is used to enable interrupts. When one of the DCMI_IER bits is set, the corresponding interrupt is enabled. This register is accessible in both read and write.

**Address offset:** 0x0C  
**Reset value:** 0x0000 0000

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

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

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

**Bit 4** **LINE_IE**: Line interrupt enable  
0: No interrupt generation when the line is received  
1: An interrupt is generated when a line has been completely received.

**Bit 3** **VSYNC_IE**: DCMI_VSYNC interrupt enable  
0: No interrupt generation  
1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state.  
The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

**Bit 2** **ERR_IE**: Synchronization error interrupt enable  
0: No interrupt generation  
1: An interrupt is generated if the embedded synchronization codes are not received in the correct order.  
*Note:* This bit is available only in embedded synchronization mode.

**Bit 1** **OVR_IE**: Overrun interrupt enable  
0: No interrupt generation  
1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received.

**Bit 0** **FRAME_IE**: Capture complete interrupt enable  
0: No interrupt generation  
1: An interrupt is generated at the end of each received frame/crop window (in crop mode).
### 30.5.5 DCMI masked interrupt status register (DCMI_MIS)

This DCMI_MIS register is a read-only register. When read, it returns the current masked status value (depending on the value in DCMI_IER) of the corresponding interrupt. A bit in this register is set if the corresponding enable bit in DCMI_IER is set and the corresponding bit in DCMI_RIS is set.

Address offset: 0x10

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:5</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 4</td>
<td><strong>LINE_MIS</strong>: Line masked interrupt status</td>
</tr>
<tr>
<td></td>
<td>This bit gives the status of the masked line interrupt.</td>
</tr>
<tr>
<td></td>
<td>0: No interrupt generation when the line is received</td>
</tr>
<tr>
<td></td>
<td>1: An interrupt is generated when a line has been completely received and the LINE_IE bit is set in DCMI_IER.</td>
</tr>
<tr>
<td>Bit 3</td>
<td><strong>VSYNC_MIS</strong>: VSYNC masked interrupt status</td>
</tr>
<tr>
<td></td>
<td>This bit gives the status of the masked VSYNC interrupt.</td>
</tr>
<tr>
<td></td>
<td>0: No interrupt is generated on DCMI_VSYNC transitions.</td>
</tr>
<tr>
<td></td>
<td>1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state and the VSYNC_IE bit is set in DCMI_IER.</td>
</tr>
<tr>
<td></td>
<td>The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.</td>
</tr>
<tr>
<td>Bit 2</td>
<td><strong>ERR_MIS</strong>: Synchronization error masked interrupt status</td>
</tr>
<tr>
<td></td>
<td>This bit gives the status of the masked synchronization error interrupt.</td>
</tr>
<tr>
<td></td>
<td>0: No interrupt is generated on a synchronization error.</td>
</tr>
<tr>
<td></td>
<td>1: An interrupt is generated if the embedded synchronization codes are not received in the correct order and the ERR_IE bit in DCMI_IER is set.</td>
</tr>
<tr>
<td></td>
<td><em>Note: This bit is available only in embedded synchronization mode.</em></td>
</tr>
<tr>
<td>Bit 1</td>
<td><strong>OVR_MIS</strong>: Overrun masked interrupt status</td>
</tr>
<tr>
<td></td>
<td>This bit gives the status of the masked overflow interrupt.</td>
</tr>
<tr>
<td></td>
<td>0: No interrupt is generated on overrun.</td>
</tr>
<tr>
<td></td>
<td>1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received and the OVR_IE bit is set in DCMI_IER.</td>
</tr>
<tr>
<td>Bit 0</td>
<td><strong>FRAME_MIS</strong>: Capture complete masked interrupt status</td>
</tr>
<tr>
<td></td>
<td>This bit gives the status of the masked capture complete interrupt.</td>
</tr>
<tr>
<td></td>
<td>0: No interrupt is generated after a complete capture.</td>
</tr>
<tr>
<td></td>
<td>1: An interrupt is generated at the end of each received frame/crop window (in crop mode) and the FRAME_IE bit is set in DCMI_IER.</td>
</tr>
</tbody>
</table>
30.5.6 DCMI interrupt clear register (DCMI_ICR)

The DCMI_ICR register is write-only. Setting a bit of this register clears the corresponding flag in the DCMI_RIS and DCMI_MIS registers. Writing 0 has no effect.

Address offset: 0x14
Reset value: 0x0000 0000

| Bit 31:5 | Reserved, must be kept at reset value. |
| Bit 4   | LINE_ISC: line interrupt status clear |
| Setting this bit clears the LINE_RIS flag in the DCMI_RIS register. |
| Bit 3   | VSYNC_ISC: Vertical Synchronization interrupt status clear |
| Setting this bit clears the VSYNC_RIS flag in the DCMI_RIS register. |
| Bit 2   | ERR_ISC: Synchronization error interrupt status clear |
| Setting this bit clears the ERR_RIS flag in the DCMI_RIS register. |
| Note: This bit is available only in embedded synchronization mode. |
| Bit 1   | OVR_ISC: Overrun interrupt status clear |
| Setting this bit clears the OVR_RIS flag in the DCMI_RIS register. |
| Bit 0   | FRAME_ISC: Capture complete interrupt status clear |
| Setting this bit clears the FRAME_RIS flag in the DCMI_RIS register. |

30.5.7 DCMI embedded synchronization code register (DCMI_ESCR)

Address offset: 0x18
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:0</th>
<th>FEC[7:0]</th>
<th>LEC[7:0]</th>
<th>LSC[7:0]</th>
<th>FSC[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Note: The bit fields are defined as read-write (rw) or read-only (ro) depending on the register state.

STMicroelectronics
Bits 31:24 **FEC[7:0]**: Frame end delimiter code

This byte specifies the code of the frame end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FEC.

If FEC is programmed to 0xFF, all the unused codes (0xFF0000XY) are interpreted as frame end delimiters.

Bits 23:16 **LEC[7:0]**: Line end delimiter code

This byte specifies the code of the line end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LEC.

Bits 15:8 **LSC[7:0]**: Line start delimiter code

This byte specifies the code of the line start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LSC.

Bits 7:0 **FSC[7:0]**: Frame start delimiter code

This byte specifies the code of the frame start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FSC.

If FSC is programmed to 0xFF, no frame start delimiter is detected. But, the first occurrence of LSC after an FEC code is interpreted as a start of frame delimiter.

---

30.5.8 **DCMI embedded synchronization unmask register (DCMI_ESUR)**

Address offset: 0x1C

Reset value: 0x0000 0000

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

**FEU[7:0]**: Frame end delimiter unmask

This byte specifies the mask to be applied to the code of the frame end delimiter.

0: The corresponding bit in the FEC byte in DCMI_ESCR is masked while comparing the frame end delimiter with the received data.

1: The corresponding bit in the FEC byte in DCMI_ESCR is compared while comparing the frame end delimiter with the received data.

**LEU[7:0]**: Line end delimiter unmask

This byte specifies the mask to be applied to the code of the line end delimiter.

0: The corresponding bit in the LEC byte in DCMI_ESCR is masked while comparing the line end delimiter with the received data.

1: The corresponding bit in the LEC byte in DCMI_ESCR is compared while comparing the line end delimiter with the received data.

**LSU[7:0]**: Line start delimiter unmask

This byte specifies the mask to be applied to the code of the line start delimiter.

0: The corresponding bit in the LSC byte in DCMI_ESCR is masked while comparing the line start delimiter with the received data.

1: The corresponding bit in the LSC byte in DCMI_ESCR is compared while comparing the line start delimiter with the received data.
Digital camera interface (DCMI)

30.5.9 DCMI crop window start (DCMI_CWSTRT)

Address offset: 0x20
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:29</th>
<th>VST[12:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:14</th>
<th>HOFFCNT[13:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 VST[12:0]: Vertical start line count
The image capture starts with this line number. Previous line data are ignored.
0x0000: line 1
0x0001: line 2
0x0002: line 3
....

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

30.5.10 DCMI crop window size (DCMI_CWSIZE)

Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:30</th>
<th>VLINE[13:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:14</th>
<th>CAPCNT[13:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:30 Reserved, must be kept at reset value.

Bits 7:0 FSU[7:0]: Frame start delimiter unmask
This byte specifies the mask to be applied to the code of the frame start delimiter.
0: The corresponding bit in the FSC byte in DCMI_ESCR is masked while comparing the frame start delimiter with the received data.
1: The corresponding bit in the FSC byte in DCMI_ESCR is compared while comparing the frame start delimiter with the received data.
30.5.11 DCMI data register (DCMI_DR)

Address offset: 0x28
Reset value: 0x0000 0000

The digital camera Interface packages all the received data in 32-bit format before requesting a DMA transfer. A 8-word deep FIFO is available to leave enough time for DMA transfers and avoid DMA overrun conditions.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 29:16 VLINE[13:0] : Vertical line count
This value gives the number of lines to be captured from the starting point.
0x0000: 1 line
0x0001: 2 lines
0x0002: 3 lines
....

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

Bits 13:0 CAPCNT[13:0] : Capture count
This value gives the number of pixel clocks to be captured from the starting point on the same line. It value must corresponds to word-aligned data for different widths of parallel interfaces.
0x0000: 1 pixel
0x0001: 2 pixels
0x0002: 3 pixels
....

30.5.12 DCMI register map

Table 288. DCMI register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | DCMI_CR      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Reset value
Refer to Section 2.3 for the register boundary addresses.
31 Parallel synchronous slave interface (PSSI)

The PSSI peripheral and the DCMI (digital camera interface) use the same circuitry. As a result, these two peripherals cannot be used at the same time: when using the PSSI, the DCMI registers cannot be accessed, and vice-versa.

In addition, the PSSI and the DCMI share the same alternate functions and interrupt vector (see Section 31.3.2: PSSI pins and internal signals).

31.1 Introduction

The PSSI is a generic synchronous 8/16-bit parallel data input/output slave interface. It enables the transmitter to send a data valid signal that indicates when the data is valid, and the receiver to output a flow control signal that indicates when it is ready to sample the data.

31.2 PSSI main features

The PSSI peripheral main features are the following:

- Slave mode operation
- 8-bit or 16-bit parallel data input or output
- 8-word (32-byte) FIFO
- Data enable (PSSI_DE) alternate function input and Ready (PSSI_RDY) alternate function output
  
  When selected, these signals can either enable the transmitter to indicate when the data is valid, allow the receiver to indicate when it is ready to sample the data, or both.

31.3 PSSI functional description

The PSSI is a synchronous parallel slave interface that can send or receive high-speed data flows. It consists of up to 16 data lines (PSSI_D[15:0]) plus a clock line (PSSI_PDCK). The clock polarity can be configured so that data can be captured or transmitted on either the clock rising or falling edge.

Usually, a general-purpose DMA channel is used to pass 32-bit packed data via the data register (PSSI_DR).

The data flow can either be continuous or synchronized by hardware using the optional PSSI_DE (Data enable), and PSSI_RDY (Ready) signals.

*Figure 281* shows the PSSI block diagram.
31.3.1 PSSI block diagram

![Figure 281. PSSI block diagram](MSv48844V3)

![Figure 282. Top-level block diagram](MSv48845V4)

31.3.2 PSSI pins and internal signals

The PSSI interface is composed of 19 pins, though nine signals are enough to transfer parallel data. Table 289 shows the PSSI pins.

When the PSSI ENABLE bit (bit 14 of PSSI_CR) is set to 1, the alternate functions and the interrupt vector are associated with the PSSI. Otherwise, they are associated with the DCMI. The DCMI ENABLE bit (bit 15 of DCMI_CR) and the PSSI ENABLE bit (bit 14 of PSSI_CR) must not be set to 1 at the same time. As an example, if a GPIO is configured to use the alternate function PSSI_PDCK/DCMI_PIXCK, it is the PSSI_PDCK function which becomes active if PSSI_CR/ENABLE is set to 1.
### 31.3.3 PSSI clock

The AHB clock frequency must be at least 2.5 times higher than the PSSI_PDCK frequency. At frequency ratios lower than 2.5, data might be corrupted or lost during transfers.

Data transfers are synchronous with PSSI_PDCK. The PSSI_PDCK polarity can be configured as follows, through CKPOL bit (bit 5 of PSSI_CR):

- When CKPOL = 0
  - Input pins are sampled on PSSI_PDCK falling edge
  - Output pins are driven on PSSI_PDCK rising edge
- When CKPOL = 1
  - Input pins are sampled on PSSI_PDCK rising edge
  - Output pins are driven on PSSI_PDCK falling edge

### 31.3.4 PSSI data management

#### Data direction

The direction of data transfers is configured through the OUTEN control bit (bit 31 of PSSI_CR):

- When OUTEN is cleared to 0 (default setting), the PSSI operates in receive mode and the data is input on the data pins.
- When OUTEN is set to 1, the peripheral operates in transmit mode and the data is output on the data pins.

OUTEN can be modified only when the ENABLE bit is cleared to 0.
Data register and DMA

Data are transferred from/to the FIFO using the PSSI_DR data register:
- In receive mode, data must be read from the FIFO by reading PSSI_DR.
- In transmit mode, data must be written to the FIFO by writing into PSSI_DR.

Word (32-bit) accesses to PSSI_DR and half-word (16-bit) accesses to PSSI_DR[15:0] are permitted in all modes. Byte (8-bit) accesses to PSSI_DR[7:0] are permitted only when the PSSI is configured to transfer 8 bits at a time (EDM=00 in the PSSI_CR register).

To reduce the load on the CPU, it is recommended to use the DMA to transfer data from/to the PSSI FIFO. When it is used, the DMA must be configured to transfer data via the PSSI_DR register. Using 32-bit transfers optimizes bandwidth and reduces the bus load. However, 8-bit and 16-bit transfers are also permitted.

To use the DMA, set the PSSI DMA enable bit (DMAEN in PSSI_CR) to 1 (default setting). When DMAEN is set to 1, a DMA transfer is initiated when the FIFO is ready for a 32-bit transfer (four valid bytes in receive mode or four empty bytes in transmit mode). As a result, in receive mode, no DMA transfers are initiated if there are three bytes or fewer in the FIFO, even if the DMA is configured to perform 8-bit transfers.

The RTT4B and RTT1B status bits (PSSI_SR) are useful when the CPU directly performs transfers to and from the FIFO. RTT4B set to 1 indicates that the FIFO is ready to transfer four bytes: at least four valid bytes in the FIFO in receive mode or at least four free bytes in transmit mode. RTT1B set to 1 indicates that the FIFO is ready to transfer one byte: at least one valid byte in the FIFO in receive mode or at least one free byte in transmit mode.

8-bit data

The PSSI parallel interface can transfer either 8-bit (using D[7:0]) or 16-bit data (using D[15:0]) depending on the EDM[1:0] control bits (bits 11:10 of PSSI_CR). If the 8-bit configuration is selected (EDM[1:0] set to 00), the unused D[15:0] pins can be used for GPIO or other functions.

When EDM[1:0] in PSSI_CR are programmed to 00, the interface transfers 8 bits using the D[7:0] pins. In this case, D[15:8] are not used and four PSSI_PDCK cycles are required to transfer a 32-bit word.

The least-significant byte (bits 7:0) correspond to the first byte transferred, and the most-significant byte (bits 31:24) corresponds to the forth byte transferred. Table 291 illustrates the positioning of the data bytes in two 32-bit words.

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:24</th>
<th>23:16</th>
<th>15:8</th>
<th>7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D_{n+3}[7:0]</td>
<td>D_{n+2}[7:0]</td>
<td>D_{n+1}[7:0]</td>
<td>D_{n}[7:0]</td>
</tr>
<tr>
<td>4</td>
<td>D_{n+7}[7:0]</td>
<td>D_{n+6}[7:0]</td>
<td>D_{n+5}[7:0]</td>
<td>D_{n+4}[7:0]</td>
</tr>
</tbody>
</table>
16-bit data
When EDM[1:0] in PSSI_CR are programmed to 11, the interface transfers 16 bits using the D[15:0] pins. In this case, two PSSI_PDCK cycles are required to transfer a 32-bit word.

The least-significant half word (bits 15:0) correspond to the first half word transferred, and the most-significant half-word (bits 31:16) corresponds to the second half word transferred. Table 292 illustrates the positioning of the data in two 32-bit words.

Table 292. Positioning of captured data bytes in 32-bit words (16-bit width)

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:16</th>
<th>15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D_{n+1}[15:0]</td>
<td>D_n[15:0]</td>
</tr>
<tr>
<td>4</td>
<td>D_{n+3}[15:0]</td>
<td>D_{n+2}[15:0]</td>
</tr>
</tbody>
</table>

FIFO data buffer and error conditions
An eight-word FIFO helps improving performance and avoids overruns and underruns.

If the ready signal (PSSI_RDY) is disabled in receive mode, an overrun error is generated when a clock active edge occurs when the FIFO is full. In this case, the input data is lost.

If the data enable signal (PSSI_DE) is disabled in transmit mode, an underrun error is generated when a clock active edge occurs when the FIFO is empty. In this case, unpredictable data are output.

The OVR_RIS status bit indicates that either an overrun or an underrun occurred. An interrupt can be generated when these events occur.

31.3.5 PSSI optional control signals

Data Enable (PSSI_DE) alternate function input
The data enable signal, PSSI_DE, is an optional signal. It is driven by the data source/transmitter in order to indicate that the data is valid to be transferred during the current cycle. When PSSI_DE is inactive, it means that the data must not be sampled by the receiver at the next clock edge.

This alternate function signal can be enabled using the DERDYCFG (bits 20:18 of PSSI_CR) control bits. PSSI_DE polarity is configured through DEPOL control bit (bit 6 of PSSI_CR). PSSI_DE is active low when DEPOL is cleared to 0, and high when DEPOL is set to 1.

The direction of the PSSI_DE signal is defined by the OUTEN value. It is the same as the data direction.

If the PSSI_DE alternate function input is enabled (through DERDYCFG) in receive mode (OUTEN cleared to 0), the PSSI samples PSSI_DE on the same PSSI_PDCK edge as the one used for sampling the data (D[15:0]). If PSSI_DE is active, the sampled data is saved in the FIFO. Otherwise, the sampled data is considered invalid and discarded. The transmitting device can use PSSI_DE as a data valid signal, driving it inactive when the data in the current cycle is not valid. This flow control function allows avoiding underrun errors.
If the PSSI_DE alternate output function is enabled (through DERDYCFG) in transmit mode (OUTEN=1), the PSSI drives PSSI_DE on the same PSSI_PDCK edge that the one used to drive the data (D[15:0]). If a new 8 or 16-bit data (as programmed in the EDM[1:0] control bits in PSSI_CR) is available for transmission in the internal FIFO, this data is output on the data outputs (D[15:0]) and the PSSI_DE output becomes active on the current PSSI_PDCK edge. Otherwise (if the TX FIFO is empty), the D[15:0] outputs remains unchanged on the next clock edge and the PSSI_DE output becomes inactive.

**Figure 284. Data enable waveform diagram in transmit mode (CKPOL=0)**

If the PSSI_RDY alternate output function is enabled (through DERDYCFG) in receive mode (OUTEN=0), the PSSI drives PSSI_RDY one PSSI_PDCK half cycle after it samples.

**Ready (PSSI_RDY) alternate function output**

The ready signal, PSSI_RDY, is an optional signal. It is driven by the receiving device and indicates whether data is being accepted in the current cycle. When PSSI_RDY is inactive, it means that the data must not be sampled by the receiver at the next clock edge.

This alternate function signal can be enabled using the DERDYCFG control bits (bits 20:18 of PSSI_CR). PSSI_RDY polarity is configured through the RDYPOL control bit (bit 6 of PSSI_CR). PSSI_RDY is active low when RDYPOL is cleared to 0, and high when RDYPOL set to 1.

The direction of the PSSI_RDY signal is defined by the OUTEN (bit 31 of PSSI_CR). It is set in the opposite direction compared to the PSSI_DE and data signals.

If the PSSI_RDY alternate output function is enabled (through DERDYCFG) in receive mode (OUTEN=0), the PSSI drives PSSI_RDY one PSSI_PDCK half cycle after it samples.
the data \((D[15:0])\). If the FIFO has enough free space to receive more data, the PSSI drives
the PSSI\_RDY signal active. Otherwise, if the FIFO is full and cannot accept more data, the
PSSI drives the PSSI\_RDY signal inactive. The transmitting device must repeat the current
data in the next cycle when it detects that PSSI\_RDY is inactive. This flow control function
allows the PSSI to avoid overrun errors when the system (via the DMA) is unable to keep up
with the data flow.

**Figure 285. Ready in receive mode waveform diagram (CKPOL=0)**

![Waveform Diagram](MSv48848V2)

If the PSSI\_RDY alternate input function is enabled (through DERDYCFG) in transmit mode
\((OUTEN=1)\), the PSSI samples the PSSI\_RDY signal on the opposite PSSI\_PDCK edge to
the one at which \(D[15:0]\) are driven. If the PSSI\_RDY signal is inactive, the PSSI keeps the
same data \((D[15:0])\) and PSSI\_DE signals that valid data are available during the next
PSSI\_PDCK clock cycle. Otherwise, if PSSI\_RDY signal is sampled as active, the next data
from the TX FIFO (if available) is output on the data outputs \((D[15:0])\). If no new data are
available in the TX FIFO, the PSSI keeps the data output values and outputs the PSSI\_DE
signal as inactive (if enabled).

The receiving device uses the PSSI\_RDY to control the data flow and avoid overrun errors
when the system (via the DMA) is unable to keep up with the data flow.

**Bidirectional PSSI\_DE/PSSI\_RDY signal**

A single pin can be used for both data enable (PSSI\_DE) and ready (PSSI\_RDY) functions
if DEPOL and RDYPOL are both set to 1 and DERDYCFG is set to 111 or 100 in the
PSSI\_CR register. In this case, the GPIO corresponding to selected alternate function
(PSSI\_DE when DERDYCFG=111 or PSSI\_RDY when DERDYCFG=100) must be
configured as open-drain. The other device must also be configured to drive the line as
open-drain, and a weak pull-up must be applied to the line.

The signal thus becomes bidirectional. If either the sender drives the line low (to indicate
that the data is not valid) or the receiver drives the line low (to indicate that it is not sampling
the current data), then both devices know that the data is not being transferred in the current
cycle.
31.4 PSSI interrupts

The PSSI generates only one interrupt (IT_OVR). It is consequently equivalent to the global interrupt (pssi_it). Refer to Table 293 for the list of interrupts.

The PSSI and the DCMI share the same interrupt vector. When the PSSI ENABLE bit (bit 14 of PSSI_CR) is set to 1, these interrupts are triggered by the PSSI. Otherwise, they are controlled by the DCMI.

The DCMI ENABLE bit (bit 14 of DCMI_CR) and PSSI ENABLE bit must not be set to 1 at the same time.

Table 293. PSSI interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Shared with DCMI</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from low-power mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT_OVR</td>
<td>IT_OVR</td>
<td>indicates overrun in receive mode or underrun in transmit mode</td>
<td>OVR_RIS</td>
<td>OVR_IE</td>
<td>OVR_ISC</td>
<td>NA</td>
</tr>
</tbody>
</table>
31.5 PSSI registers

An 8-bit write or a 16-bit write operation to any PSSI register besides PSSI_DR, results in a bus error. 32-bit read and write operations are permitted.

31.5.1 PSSI control register (PSSI_CR)

Address offset: 0x00
Reset value: 0x4000 0000

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

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

Bit 31 OUTEN: Data direction selection bit
0: Receive mode: data is input synchronously with PSSI_PDCK
1: Transmit mode: data is output synchronously with PSSI_PDCK

Bit 30 DMAEN: DMA enable bit
0: DMA transfers are disabled. The user application can directly access the PSSI_DR register when DMA transfers are disabled.
1: DMA transfers are enabled (default configuration). A DMA channel in the general-purpose DMA controller must be configured to perform transfers from/to PSSI_DR.

Bits 29:21 Reserved, must be kept at reset value.

Bits 20:18 DERDYCFG[2:0]: Data enable and ready configuration
000: PSSI_DE and PSSI_RDY both disabled
001: Only PSSI_RDY enabled
010: Only PSSI_DE enabled
011: Both PSSI_RDY and PSSI_DE alternate functions enabled
100: Both PSSI_RDY and PSSI_DE features enabled - bidirectional on PSSI_RDY pin (see Bidirectional PSSI_DE/PSSI_RDY signal on page 1225)
101: Only PSSI_RDY function enabled, but mapped to PSSI_DE pin
110: Only PSSI_DE function enabled, but mapped to PSSI_RDY pin
111: Both PSSI_RDY and PSSI_DE features enabled - bidirectional on PSSI_DE pin (see Bidirectional PSSI_DE/PSSI_RDY signal on page 1225)

When the PSSI_RDY function is mapped to the PSSI_DE pin (settings 101 or 111), it is still the RDYPOL bit which determines its polarity. Similarly, when the PSSI_DE function is mapped to the PSSI_RDY pin (settings 110 or 111), it is still the DEPOL bit which determines its polarity.

Bits 17:15 Reserved, must be kept at reset value.
31.5.2 PSSI status register (PSSI_SR)

Address offset: 0x04

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 14 ENABLE: PSSI enable
0: PSSI disabled
1: PSSI enabled
The contents of the FIFO are flushed when ENABLE is cleared to 0.

Note: When ENABLE=1, the content of PSSI_CR must not be changed, except for the ENABLE bit itself. All configuration bits can change as soon as ENABLE changes from 0 to 1.
The DMA controller and all PSSI configuration registers must be programmed correctly before setting the ENABLE bit to 1.
The ENABLE bit and the DCMI ENABLE bit (bit 15 of DCMI_CR) must not be set to 1 at the same time.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:10 EDM[1:0]: Extended data mode
00: Interface captures 8-bit data on every parallel data clock
01: Reserved, must not be selected
10: Reserved, must not be selected
11: The interface captures 16-bit data on every parallel data clock

Bits 9 Reserved, must be kept at reset value.

Bit 8 RDYPOL: Ready (PSSI_RDY) polarity
This bit indicates the level on the PSSI_RDY pin when the data are not valid on the parallel interface.
0: PSSI_RDY active low (0 indicates that the receiver is ready to receive)
1: PSSI_RDY active high (1 indicates that the receiver is ready to receive)

Bits 7 Reserved, must be kept at reset value.

Bit 6 DEPOL: Data enable (PSSI_DE) polarity
This bit indicates the level on the PSSI_DE pin when the data are not valid on the parallel interface.
0: PSSI_DE active low (0 indicates that data is valid)
1: PSSI_DE active high (1 indicates that data is valid)

Bit 5 CKPOL: Parallel data clock polarity
This bit configures the capture edge of the parallel clock or the edge used for driving outputs, depending on OUTEN.
0: Falling edge active for inputs or rising edge active for outputs
1: Rising edge active for inputs or falling edge active for outputs.

Bits 4:0 Reserved, must be kept at reset value.
31.5.3 PSSI raw interrupt status register (PSSI_RIS)

Address offset: 0x08
Reset value: 0x0000 0000

PSSI_RIS gives the raw interrupt status. This register is read-only. When read, it returns the status of the corresponding interrupt before masking with the PSSI_IER register value.

| Bits 31:2 | Reserved, must be kept at reset value. |
| 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 |
| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
|           r               |

Bit 3 RTT1B: FIFO is ready to transfer one byte
1: FIFO is ready for a one byte (32-bit) transfer. In receive mode, this means that at least one valid data byte is in the FIFO. In transmit mode, this means that there is at least one byte free in the FIFO.
0: FIFO is not ready for a 1-byte transfer

Bit 2 RTT4B: FIFO is ready to transfer four bytes
1: FIFO is ready for a four-byte (32-bit) transfer. In receive mode, this means that at least four valid data bytes are in the FIFO. In transmit mode, this means that there are at least four bytes free in the FIFO.
0: FIFO is not ready for a four-byte transfer

Bits 1:0 Reserved, must be kept at reset value.
### 31.5.4 PSSI interrupt enable register (PSSI_IER)

Address offset: 0x0C  
Reset value: 0x0000 0000

The PSSI_IER register is used to enable interrupts. When one of the PSSI_IER bits is set, the corresponding interrupt is enabled. This register is accessible both in read and write modes.

<table>
<thead>
<tr>
<th>Address offset</th>
<th>Reset value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0C</td>
<td>0x0000 0000</td>
<td>PSSI_IER register</td>
</tr>
</tbody>
</table>

#### Bits 31:2

- **Bit 0** Reserved, must be kept at reset value.
- **Bit 1** OVR_IE: Data buffer overrun/underrun interrupt enable  
  0: No interrupt generation  
  1: An interrupt is generated if either an overrun or an underrun error occurred.

#### Bits 15:2

- **Bit 1** OVR_MIS: Data buffer overrun/underrun masked interrupt status  
  This bit is set to 1 only when PSSI_IER/OVR_IE and PSSI_RIS/OVR_RIS are both set to 1.  
  0: No interrupt is generated when an overrun/underrun error occurs  
  1: An interrupt is generated if there is either an overrun or an underrun error and the OVR_IE bit is set in PSSI_IER.

- **Bit 0** Reserved, must be kept at reset value.

---

### 31.5.5 PSSI masked interrupt status register (PSSI_MIS)

This PSSI_MIS register is read-only. When read, it returns the current masked status value of the corresponding interrupt (depending on the value in PSSI_IER). A bit in this register is set if the corresponding enable bit in PSSI_IER is set and the corresponding bit in PSSI_RIS is set.

Address offset: 0x10  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address offset</th>
<th>Reset value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10</td>
<td>0x0000 0000</td>
<td>PSSI_MIS register</td>
</tr>
</tbody>
</table>

#### Bits 31:2

- **Bit 0** Reserved, must be kept at reset value.
- **Bit 1** OVR_MIS: Data buffer overrun/underrun masked interrupt status  
  This bit is set to 1 only when PSSI_IER/OVR_IE and PSSI_RIS/OVR_RIS are both set to 1.  
  0: No interrupt is generated when an overrun/underrun error occurs  
  1: An interrupt is generated if there is either an overrun or an underrun error and the OVR_IE bit is set in PSSI_IER.

- **Bit 0** Reserved, must be kept at reset value.
31.5.6 PSSI interrupt clear register (PSSI_ICR)

Address offset: 0x14
Reset value: 0x0000 0000

The PSSI_ICR register is write-only. Writing a 1 into a bit of this register clears the corresponding bit in the PSSI_RIS and PSSI_MIS registers. Writing a 0 has no effect. Reading this register always gives zeros.

31.5.7 PSSI data register (PSSI_DR)

Address offset: 0x28
Reset value: 0x0000 0000

In receive mode (OUTEN = 0), the DMA controller must read the received data from this register. Write operations to PSSI_DR result in an error response. When more bytes than the number of valid bytes are read in the FIFO, the invalid bytes return zeros.

In transmit mode (OUTEN = 1), the DMA controller must write the data to be transmitted into this register. Read operations to PSSI_DR result in an error response.

32-bit, 16-bit, and 8-bit accesses are all supported for PSSI_DR. For instance, 16-bit read/write operations remove/add two bytes from/to the FIFO. However, 8-bit accesses are permitted only when the PSSI is configured to transfer 8 data bits at a time (EDM=00 in PSSI_CR). 8-bit accesses to PSSI_DR when EDM is not set to 0 result in an error response.

All accesses must include byte 0: 8-bit accesses must be performed to bits 7 to 0 and 16-bit accesses from bits 15 to 0. Accesses that do not include byte 0 results in an error response.

Accessing PSSI_DR when ENABLE bit in PSSI_CR is set to 0 results in an error response.
31.5.8 PSSI register map

| Offset | Register name reset value | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|--------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | PSSI_CR                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | OUTEN                    | 0  | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DMAEN                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DERDCFG                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | ENABLE[2:0]              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | EDM                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | RDYPOL                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DEPOL                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | OPOL                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | PSSI_SR                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | PSSI_RIS                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C   | PSSI_IER                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10   | PSSI_MIS                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x14   | PSSI_ICR                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x18-  | Reserved                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x20   | PSSI_DR                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | BYTE3[7:0]               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | BYTE2[7:0]               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | BYTE1[7:0]               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | BYTE0[7:0]               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3 for the register boundary addresses.
32 True random number generator (RNG)

32.1 Introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG is a NIST SP 800-90B compliant entropy source that can be used to construct a nondeterministic random bit generator (NDRBG).

The RNG true random number generator has been precertified NIST SP800-90B. It has also been tested using the German BSI statistical tests of AIS-31 (T0 to T8).

32.2 RNG main features

- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by a NIST SP800-90B approved conditioning stage.
- It can be used as the entropy source to construct a nondeterministic random bit generator (NDRBG).
- In the NIST configuration, it produces four 32-bit random samples every 412 AHB clock cycles if \( f_{\text{AHB}} < f_{\text{threshold}} \) (256 RNG clock cycles otherwise).
- It embeds startup and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management.
- It can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- It has an AMBA® AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).
32.3  RNG functional description

32.3.1  RNG block diagram

*Figure 288* shows the RNG block diagram.

![RNG block diagram](image)

32.3.2  RNG internal signals

*Table 295* describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rng_it</td>
<td>Digital output</td>
<td>RNG global interrupt request</td>
</tr>
<tr>
<td>rng_hclk</td>
<td>Digital input</td>
<td>AHB clock</td>
</tr>
<tr>
<td>rng_clk</td>
<td>Digital input</td>
<td>RNG dedicated clock, asynchronous to rng_hclk</td>
</tr>
<tr>
<td>rng_itamp_out</td>
<td>Digital output</td>
<td>RNG internal tamper event signal to TAMP (XORed), triggered when an unexpected hardware fault occurs. When this signal is triggered, RNG stops delivering random samples, requiring a reset and a new initialization to be usable again.</td>
</tr>
</tbody>
</table>
32.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary RNG integrates all the required NIST components depicted on Figure 289. Those components are an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

![Figure 289. NIST SP800-90B entropy source model](image)

The components pictured above are detailed hereafter.

Noise source

The noise source is the component that contains the non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstring output by the entropy source. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x6), each based on three XORed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in Section 32.3.8: RNG low-power use.
- The XORing of all the noise sources into a single analog output.
- A sampling stage of this output clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (rng_hclk), with a possibility for the software to decrease the sampling frequency by using the integrated divider.

Note: In Section 32.6: RNG entropy source validation the recommended RNG clock frequencies and associated divider value are given.
Post processing

In the NIST configuration no post-processing is applied to the sampled noise source. In non-NIST configuration B (as defined in Section 32.6.2) a normalization debiasing is applied, that is half of the bits are taken from the sampled noise source, half of the bits are taken from the inverted sampled noise source.

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bitstrings output (128-bit). The NIST SP800-90B target is full entropy on the output (128-bit).

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in Section 32.5: RNG processing time.

Output buffer

A data output buffer can store up to four 32-bit words that have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register after a number of clock cycles specified in Section 32.5: RNG processing time.

Whenever a random number is available through the RNG_DR register, the DRDY flag changes from 0 to 1. This flag remains high until the output buffer becomes empty after reading four words from the RNG_DR register.

Note: When interrupts are enabled an interrupt is generated when this data ready flag transitions from 0 to 1. Interrupt is then cleared automatically by the RNG as explained above.
Health checks

This component ensures that the entire entropy source (with its noise source) starts then operates as expected, obtaining assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the value recommended for register RNG_HTCR (configuration A in Section 32.6.2).

1. Startup health tests, performed after reset and before the first use of the RNG as entropy source:
   - Repetition count test, flagging an error when the noise source has provided more than 42 consecutive bits at a constant value (0 or 1).
   - Adaptive proportion test running on a window of 1024 consecutive bits: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 711 times.
   - Known-answer tests, to verify the conditioning stage.

2. Continuous health tests, running indefinitely on the outputs of the noise source:
   - Repetition count test, similar to the one running in startup tests.
   - Adaptive proportion test, similar to the one running in startup tests.

3. Vendor specific continuous tests
   - Transition count test, flagging an error when the noise source has delivered more than 32 consecutive occurrences of 2-bit patterns (01 or 10).
   - Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than AHB clock cycle divided by 32.

4. On-demand test of digitized noise source (raw data)
   - Supported by restarting the entropy source and rerunning the startup tests (see software reset sequence in Section 32.3.4: RNG initialization). Other kinds of on-demand testing (software based) are not supported.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in Section 32.3.7: Error management.

Note: An interrupt can be generated when an error is detected.

Above the health test thresholds are modified by changing the value in the RNG_HTCR register. See Section 32.6: RNG entropy source validation for details.
32.3.4 RNG initialization

The RNG simplified state machine is pictured on Figure 290.

After enabling the RNG (RNGEN = 1 in RNG_CR), the following chain of events occurs:

1. The analog noise source is enabled, and by default the RNG waits 16 cycles of RNG clock cycles (before divider) before starting to sample the analog output and filling the 128-bit conditioning shift register.
2. The conditioning hardware initializes, automatically triggering startup behavior test on the raw data samples and known-answer tests.
3. When startup health tests are completed. During this time, three 128-bit noise source samples are used.
4. The conditioning stage internal input data buffer is filled again with 128-bit and a number of conditioning rounds defined by the RNG configuration (NIST or non-NIST) is performed. The output buffer is then filled with the post processing result.
5. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in Section 32.5: RNG processing time.

Figure 290. RNG initialization overview
Figure 290 also highlights a possible software reset sequence, implemented by:

1. Writing bits RNGEN = 0 and CONDRST = 1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.
2. Then writing RNGEN = 1 and CONDRST = 0 in the RNG_CR register.
3. Wait for random number to be ready, after initialization completes.

**Note:** When the RNG peripheral is reset through RCC (hardware reset), the RNG configuration for optimal randomness is lost in the RNG registers. Software reset with CONFIGLOCK set preserves the RNG configuration.

### 32.3.5 RNG operation

**Normal operations**

To run the RNG using interrupts, the following steps are recommended:

1. Consult Section 32.6: RNG entropy source validation and verify if a specific RNG configuration is required for the application.
   - If it is the case, write in the RNG_CR register the bit CONDRST = 1 together with the correct RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST = 0, the interrupt enable bit IE = 1 and the RNG enable bit RNGEN = 1.
   - If it is not the case perform a write to the RNG_CR register with the interrupt enable bit IE = 1 and the RNG enable bit RNGEN = 1.
2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore, at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   - If the above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data is available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in Section 32.3.7 must be used.

To run the RNG in polling mode following steps are recommended:

1. Consult Section 32.6: RNG entropy source validation and verify if a specific RNG configuration is required for the application.
   - If it is the case write in the RNG_CR register the bit CONDRST = 1 together with the correction RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST = 0 and the RNG enable bit RNGEN = 1.
   - If it is not the case only enable the RNG by setting the RNGEN bit to 1 in the RNG_CR register.
2. Read the RNG_SR register and check that:
   - No error occurred (the SEIS and CEIS bits must be set to 0)
   - A random number is ready (the DRDY bit must be set to 1)
3. If above conditions are true read the content of the RNG_DR register up to four consecutive times. If valid data is available in the conditioning output buffer four
additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in Section 32.3.7 must be used.

**Note:** When data is not ready (DRDY = 0) RNG_DR returns zero. It is recommended to always verify that RNG_DR is different from zero. Because when it is the case a seed error occurred between RNG_SR polling and RND_DR output reading (rare event).

If the random number generation period is a concern to the application and if NIST compliance is not required it is possible to select a faster RNG configuration by using the RNG configuration “B”, described in Section 32.6: RNG entropy source validation. The gain in random number generation speed is summarized in Section 32.5: RNG processing time.

**Low-power operations**

If the power consumption is a concern to the application, low-power strategies can be used, as described in Section 32.3.8: RNG low-power use.

**Software post-processing**

No specific software post-processing/conditioning is expected to meet the AIS-31 or NIST SP800-90B approvals.

Built-in health check functions are described in Section 32.3.3: Random number generation.

### 32.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock. The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock, coupled with a programmable divider (see CLKDIV bitfield in the RNG_CR register) is used for noise source sampling. Recommended clock configurations are detailed in Section 32.6: RNG entropy source validation.

**Note:** When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency before the internal divider must be higher than the AHB clock frequency divided by 32, otherwise the clock checker always flags a clock error (CECS = 1 in the RNG_SR register).

See Section 32.3.1: RNG block diagram for details (AHB and RNG clock domains).

### 32.3.7 Error management

In parallel to random number generation a health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

**Clock error detection**

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see Section 32.3.6: RNG clocking) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.
Note: The clock error has no impact on generated random numbers that is the application can still read the RNG_DR register.  

CEIS is set only when CECS is set to 1 by RNG.

Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred. If a value is available in the RNG_DR register, it must not be used as it may not have enough entropy.

The following sequence must be used to fully recover from a seed error:

1. Software reset by writing CONDRST at 1 and at 0 (see bitfield description for details). This step is needed only if SECS is set. Indeed, when SEIS is set and SECS is cleared it means RNG performed the reset automatically (auto-reset). In this case application must clear the SEIS bit interrupt flag.
2. If SECS was set in step 1 (no auto-reset) wait for CONDRST to be cleared in the RNG_CR register, then confirm that SEIS is cleared in the RNG_SR register. Otherwise, just clear the SEIS bit in the RNG_SR register.
3. If SECS was set in step 1 (no auto-reset), wait for SECS to be cleared by RNG. The random number generation is now back to normal.

Note: After a seed error RNG restarts generating random numbers when SECS is cleared. When the application sets the ARDIS bit in the RNG_CR register, the auto-reset is disabled. CONDRST must be used in step 1.

RNG tamper errors

When an unexpected error is found by the RNG an internal tamper event is triggered in the TAMP peripheral, and the RNG stops delivering random data.

When this event occurs, the secure application needs to reset the RNG peripheral either using the central reset management or the global SoC reset. Then a proper initialization of the RNG is required, again.

32.3.8 RNG low-power use

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG_CR register. As the post-processing logic and the output buffer remain operational while RNGEN = 0 following features are available to the software:

- If there are valid words in the output buffer four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register four additional random numbers can be still be read from the RNG_DR register. If it is not the case RNG must be reenabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode) and a complete conditioning round is done. Four new random words are then available only if the expected number of conditioning round is reached (two if NISTC = 0). The overall time can be found in Section 32.5: RNG processing time on page 1243.

When disabling the RNG the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates
all the logic clocked by the RNG clock. Note that this strategy is adding latency before a 
random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the 
first time), the initialization sequence resumes from where it was stopped when RNGEN bit 
is set to 1, unless the application resets the conditioning logic using CONDRST bit in the 
RNG_CR register.

When the application wants to gate the RNG clock it is recommended to wait two RNG 
kernel clock cycles between clearing the RNGEN bit and gating the RNG kernel clock using 
the RCC.

Also, when application needs to enter a power mode where RNG is de-activated, it is 
recommended to wait two RNG kernel clock cycles between clearing the RNGEN bit and 
entering the low power mode using the PWR.

In the two cases above, to avoid unexpected consumption when RNG analog oscillators 
stay active, application can set the bit 13 in RNG_CR register. Setting this bit adds some 
marginal power consumption while RNGEN bit is set (RNG activated).

Note: The power modes where RNG is deactivated (that is retained or not available) can be found 
in the PWR section.

32.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:
- Data ready flag
- Seed error, see Section 32.3.7: Error management
- Clock error, see Section 32.3.7: Error management

Dedicated interrupt enable control bits are available as shown in Table 296.

Table 296. RNG interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNG</td>
<td>Data ready flag</td>
<td>DRDY</td>
<td>IE</td>
<td>None (automatic)</td>
</tr>
<tr>
<td></td>
<td>Seed error flag</td>
<td>SEIS</td>
<td>IE</td>
<td>Write 0 to SEIS or write CONDRST to 1 then to 0</td>
</tr>
<tr>
<td></td>
<td>Clock error flag</td>
<td>CEIS</td>
<td>IE</td>
<td>Write 0 to CEIS</td>
</tr>
</tbody>
</table>

The user can enable or disable the above interrupt sources individually by changing the 
mask bits or the general interrupt control bit IE in the RNG_CR register. The status of the 
individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.
32.5 **RNG processing time**

In recommended configuration A described in Table 297, the time between two sets of four 32-bit data is either:

- 206 x N AHB cycles if $f_{AHB} < f_{\text{threshold}}$ (conditioning stage is limiting), or
- 128 x N RNG cycles $f_{AHB} \geq f_{\text{threshold}}$ (noise source stage is limiting).

With $f_{\text{threshold}} = 1.6 \times f_{\text{RNG}}$, for instance 77 MHz if $f_{\text{RNG}} = 48$ MHz. Value N is 2.

**Note:** When CLKDIV is different from zero, $f_{\text{RNG}}$ must take into account the internal divider ratio.

If configuration B is selected the performance figures become:

- 206 AHB cycles if $f_{AHB} < f_{\text{threshold}}$ or
- 32 RNG cycles $f_{AHB} \geq f_{\text{threshold}}$

with $f_{\text{threshold}} = 6.5 \times f_{\text{RNG}}$.

32.6 **RNG entropy source validation**

32.6.1 **Introduction**

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral using the German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite. The results can be provided on demand or the customer can reproduce the tests.

32.6.2 **Validation conditions**

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock $\text{rng}_{\text{clk}} = 48$ MHz
- RNG configurations described in Table 297: RNG configurations. Note that only configuration A can be certified NIST SP800-90B.

<table>
<thead>
<tr>
<th>Table 297. RNG configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RNG Config.</strong></td>
</tr>
<tr>
<td><strong>NISTC bit</strong></td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
</tbody>
</table>

1. For NIST certification the noise source sampling must be 48 MHz or less. Hence, if the RNG clock is different from 48 MHz, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in Section 32.7.1 for details.
2. 0x1 value is recommended when RNG power consumption is critical. See the end of Section 32.3.8: RNG low-power use for details.
3. Corresponds to 42 for repetition tests and 711 for adaptive tests. See Health checks on page 1237 for details.
32.6.3 Data collection

In order to run statistical tests, it is required to collect samples from the entropy source at the raw data level as well as at the output of the entropy source. For details on data collection and the running of statistical test suites refer to “STM32 microcontrollers random number generation validation using NIST statistical test suite” application note (AN4230) available from www.st.com.

Contact STMicroelectronics if the above samples need to be retrieved for the product.

32.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

32.7.1 RNG control register (RNG_CR)

Address offset: 0x000
Reset value: 0x0080 0D00

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>CONFGLOCK: RNG Config lock</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: Writes to the RNG_HTCR and RNG_CR configuration bits [29:4] are allowed.</td>
</tr>
<tr>
<td></td>
<td>1: Writes to the RNG_HTCR and RNG_CR configuration bits [29:4] are ignored until the next RNG reset.</td>
</tr>
<tr>
<td></td>
<td>This bitfield is set once: if this bit is set it can only be reset to 0 if RNG is reset.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>CONDRST: Conditioning soft reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_CR and RNG_HTCR are not changed by CONDRST.</td>
</tr>
<tr>
<td></td>
<td>This bit must be set to 1 in the same access that set any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.</td>
</tr>
<tr>
<td></td>
<td>When CONDRST is set to 0 by the software, its value goes to 0 when the reset process is done. It takes about 2 AHB clock cycles + 2 RNG clock cycles.</td>
</tr>
</tbody>
</table>

| Bits 29:26 | Reserved, must be kept at reset value. |
| Bits 25:20 | RNG_CONFIG1[5:0]: RNG configuration 1 |
|           | Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in Section 32.6: RNG entropy source validation. |
|           | Writing any bit of RNG_CONFIG1 is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFGLOCK remains at 0. Writing to this bit is ignored if CONFGLOCK = 1. |
Bits 19:16 **CLKDIV[3:0]:** Clock divider factor
This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN = 0).
0x0: Internal RNG clock after divider is similar to incoming RNG clock.
0x1: Two RNG clock cycles per internal RNG clock.
0x2: \(2^2 \approx 4\) RNG clock cycles per internal RNG clock.
... 
0xF: \(2^{15}\) RNG clock cycles per internal clock (for example, an incoming 48 MHz RNG clock becomes a 1.5 kHz internal RNG clock)
Writing these bits is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 15:13 **RNG_CONFIG2[2:0]:** RNG configuration 2
Reserved to the RNG configuration (bitfield 2). Bit 13 can be set when RNG power consumption is critical. See Section 32.3.8: RNG low-power use. Refer to the RNG_CONFIG1 bitfield for details.

Bit 12 **NISTC:** NIST custom
0: Hardware default values for NIST compliant RNG. In this configuration per 128-bit output two conditioning loops are performed and 256 bits of noise source are used.
1: Custom values for NIST compliant RNG. See Section 32.6: RNG entropy source validation for proposed configuration.
Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 11:8 **RNG_CONFIG3[3:0]:** RNG configuration 3
Reserved to the RNG configuration (bitfield 3). Refer to RNG_CONFIG1 bitfield for details. If the NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

Bit 7 **ARDIS:** Auto reset disable
0: When a noise source error occurs RNG performs an automatic reset to clear the SECS bit.
1: When a noise source error occurs the application must reset RNG by writing CONDRST to 1 then to 0, in order to restart random number generation.
When auto-reset is enabled the application still need to clear the SEIS bit after a noise source error.
Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection
0: Clock error detection enabled
1: Clock error detection is disabled
The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED, the RNG must be disabled.
Writing this bit is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt enable
0: RNG interrupt is disabled
1: RNG interrupt is enabled. An interrupt is pending as soon as DRDY = 1, SEIS = 1 or CEIS = 1 in the RNG_SR register.
Bit 2 **RNGEN**: True random number generator enable

0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.

1: True random number generator is enabled.

Bits 1:0 Reserved, must be kept at reset value.
32.7.2 RNG status register (RNG_SR)

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rc_w0</td>
<td>rc_w0</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 SEIS: Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing 0 (unless CONDRST is used). Writing 1 has no effect.

0: No faulty sequence detected
1: At least one faulty sequence is detected. See SECS bit description for details.

An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 CEIS: Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.

0: The RNG clock is correct (f_{RNGCLK} > f_{HCLK}/32)
1: The RNG clock before the internal divider is detected too slow (f_{RNGCLK} < f_{HCLK}/32)

An interrupt is pending if IE = 1 in the RNG_CR register.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 SECS: Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.
1: At least one of the following faulty sequences has been detected:
   - Runtime repetition count test failed (noise source has provided more than 24 consecutive bits at a constant value 0 or 1, or more than 32 consecutive occurrence of two bits patterns 01 or 10)
   - Startup or continuous adaptive proportion test on noise source failed.
   - Startup post-processing/conditioning sanity check failed.

Bit 1 CECS: Clock error current status

0: The RNG clock is correct (f_{RNGCLK} > f_{HCLK}/32). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.
1: The RNG clock is too slow (f_{RNGCLK} < f_{HCLK}/32).

Note: CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 DRDY: Data ready

0: The RNG_DR register is not yet valid, no random data is available.
1: The RNG_DR register contains valid random data.

Once the output buffer becomes empty (after reading the RNG_DR register), this bit returns to 0 until a new random value is generated.

Note: The DRDY bit can rise when the peripheral is disabled (RNGEN = 0 in the RNG_CR register).

If IE=1 in the RNG_CR register, an interrupt is generated when DRDY = 1.
32.7.3 RNG data register (RNG_DR)

Address offset: 0x008
Reset value: 0x0000 0000

The RNG_DR register is a read-only register that delivers a 32-bit random value when read. The content of this register is valid when the DRDY = 1 and the value is not 0x0, even if RNGEN = 0.

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

Bits 31:0 RNDATA[31:0]: Random data
32-bit random data, which are valid when DRDY = 1. When DRDY = 0, the RNDATA value is zero.
When DRDY is set, it is recommended to always verify that RNG_DR is different from zero. Because when it is the case a seed error occurred between RNG_SR polling and RND_DR output reading (rare event).

32.7.4 RNG health test control register (RNG_HTCR)

Address offset: 0x010
Reset value: 0x0000 72AC

Writing in RNG_HTCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in the RNG_CR. Writing to this register is ignored if CONFIGLOCK=1.

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

Bits 31:0 HTCFCFG[31:0]: health test configuration
This configuration is used by RNG to configure the health tests. See Section 32.6: RNG entropy source validation for the recommended value.
Note: The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.
### 32.7.5 RNG register map

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | RNG_CR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |              | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x004  | RNG_SR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |              | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x008  | RNG_DR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | RNDATA[31:0] | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x010  | RNG_HTCR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | HTCFG[31:0]  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 0  | 0  | 1  | 0  | 1  | 0  | 1  | 0  | 1  | 0  |
|        |              | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 0  | 0  | 1  | 0  | 1  | 0  | 1  | 0  | 1  |

Refer to Section 2.3 for the register boundary addresses.
33 **AES hardware accelerator (AES)**

### 33.1 Introduction

The AES hardware accelerator (AES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST.

AES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128 or 256 bits. AES has the possibility to load by hardware the key stored in SAES peripheral, under SAES control.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required).

### 33.2 AES main features

- Compliant with NIST FIPS publication 197 “*Advanced encryption standard (AES)*” (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 51 or 75 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Using dedicated key bus, optional key sharing with side-channel resistant SAES peripheral (Shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.
- Possibility for software to suspend a message if AES needs to process another message with a higher priority, then resume the original message
33.3 AES implementation

The devices have one AES peripheral, implemented as per the following table. It can use the key generated by the SAES peripheral. For comparison, the SAES peripheral is also included in the table.

<table>
<thead>
<tr>
<th>Modes or features(1)</th>
<th>AES</th>
<th>SAES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECB, CBC chaining</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CTR, CCM, GCM chaining</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>AES 128-bit ECB encryption in cycles</td>
<td>51</td>
<td>480</td>
</tr>
<tr>
<td>DHUK and BHK key selection</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Resistance to side-channel attacks</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Shared key between SAES and AES</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Key sizes in bits</td>
<td>128, 256</td>
<td>128, 256</td>
</tr>
</tbody>
</table>

1. X = supported.

33.4 AES functional description

33.4.1 AES block diagram

*Figure 291* shows the block diagram of AES.

*Figure 291. AES block diagram*
33.4.2 AES internal signals

*Table 300* describes the user relevant internal signals interfacing the AES peripheral.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aes_hclk</td>
<td>Input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>aes_it</td>
<td>Output</td>
<td>AES interrupt request</td>
</tr>
<tr>
<td>aes_in_dma</td>
<td>Input/Output</td>
<td>AES incoming data DMA single request/acknowledge</td>
</tr>
<tr>
<td>aes_out_dma</td>
<td>Input/Output</td>
<td>AES processed data DMA single request/acknowledge</td>
</tr>
<tr>
<td>aes_itamp_out</td>
<td>Output</td>
<td>Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, AES automatically clears key registers. A reset is required for AES to be usable again.</td>
</tr>
</tbody>
</table>

33.4.3 AES reset and clocks

The AES peripheral is clocked by the AHB bus clock. The AES has a dedicated reset bit in the RCC.

33.4.4 AES symmetric cipher implementation

The AES hardware accelerator (AES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 301*. Those functions can be certified NIST PUB 140-3.

<table>
<thead>
<tr>
<th>Operations</th>
<th>Algorithm</th>
<th>Specification</th>
<th>Key bit lengths</th>
<th>Chaining modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption, decryption</td>
<td>AES</td>
<td>FIPS PUB 197, NIST SP800-38A</td>
<td>128, 256</td>
<td>ECB, CBC, CTR</td>
</tr>
<tr>
<td>Authenticated encryption or decryption</td>
<td>AES</td>
<td>NIST SP800-38C, NIST SP800-38D</td>
<td></td>
<td>GCM, CCM</td>
</tr>
<tr>
<td>Cipher-based message authentication code</td>
<td>AES</td>
<td>NIST SP800-38D</td>
<td></td>
<td>GMAC</td>
</tr>
</tbody>
</table>

AES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext). It is possible to suspend then resume any AES processing, following the sequence described in Section 33.4.8.
33.4.5 AES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 292. Encryption/decryption typical usage

Initialization

The AES peripheral is initialized according to the chaining mode. Refer to Section 33.4.9: AES basic chaining modes (ECB, CBC) and Section 33.4.10: AES counter (CTR) mode for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to Section 33.4.7: AES ciphertext stealing and data padding if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. When KEYVALID is set, enable the AES peripheral, by setting the EN bit of the AES_CR register (if not already done).
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the AES_DINR register.
   b) Wait until the status flag CCF is set in the AES_ISR register, then read the four data words from the AES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
   d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using...
Note: Up to three wait cycles are automatically inserted between two consecutive writes to the AES_DINR register, to allow sending the key to the AES processor. NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

The method uses interrupt from the AES peripheral to control the data append, through the following sequence:

1. Enable interrupts from AES, by setting the CCFIE bit of the AES_IER register.
2. When KEYVALID is set, enable the AES peripheral, by setting EN (if not already done).
3. Write first four input data words into the AES_DINR register.
4. Handle the data in the AES interrupt service routine. Upon each interrupt:
   a) Read four output data words from the AES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the AES_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the AES peripheral by clearing EN and quit the interrupt service routine.
   e) Write next four input data words into the AES_DINR register and quit the interrupt service routine.

Note: AES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two AES computations. The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

Appending data using DMA

With this method, all the transfers and processing are managed by DMA and AES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the AES peripheral input and the processed data from the AES peripheral output to the memory, as described in Section 33.6: AES DMA requests. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer must not include the last four-word block if padded with zeros. The sequence described in Appending data using the CPU in polling mode must be used instead for this last block, because the
NPBLB[3:0] bitfield must be set up before processing the block, for AES to compute a correct tag.

3. When KEYVALID is set, enable the AES peripheral, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the transfer completion, get the AES-processed data from the memory.

**Note:** The CCF flag has no use with this method because the reading of the AES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

### 33.4.6 AES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

![Figure 293. Typical operation with authentication](image)

Section 33.4.11: AES Galois/counter mode (GCM) and Section 33.4.13: AES counter with CBC-MAC (CCM) describe detailed sequences supported by AES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by AES is described in Section 33.4.12: AES Galois message authentication code (GMAC).

### 33.4.7 AES ciphertext stealing and data padding

When using AES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques such as those described in NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode. Since
AES does not implement such techniques, the application must complete the last block of input data using data from the second last block.

*Note:* Ciphertext stealing techniques are not documented in this reference manual.

Similarly, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since AES does not implement automatic data padding operation to the last block, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

### 33.4.8 AES suspend and resume operations

A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as AES is enabled again, to receive a next data block.

*Figure 294* gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

*Figure 294. Example of suspend mode management*

A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

### 33.4.9 AES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB, a special key scheduling is required before processing the first block.
Figure 295 and Figure 296 describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.

**Figure 295. ECB encryption**

<table>
<thead>
<tr>
<th>Block 1</th>
<th>Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIN (plaintext P1)</td>
<td>DIN (plaintext P2)</td>
</tr>
<tr>
<td>DATATYPE[1:0] →</td>
<td>DATATYPE[1:0] →</td>
</tr>
<tr>
<td>Swap management</td>
<td>Swap management</td>
</tr>
<tr>
<td>Encrypt</td>
<td>Encrypt</td>
</tr>
<tr>
<td>DATATYPE[1:0] ←</td>
<td>DATATYPE[1:0] ←</td>
</tr>
<tr>
<td>Swap management</td>
<td>Swap management</td>
</tr>
<tr>
<td>O1</td>
<td>O2</td>
</tr>
<tr>
<td>DOUT (ciphertext C1)</td>
<td>DOUT (ciphertext C2)</td>
</tr>
</tbody>
</table>

**Legend**

- input
- output

**Figure 296. ECB decryption**

<table>
<thead>
<tr>
<th>Block 1</th>
<th>Block 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIN (ciphertext C1)</td>
<td>DIN (ciphertext C2)</td>
</tr>
<tr>
<td>DATATYPE[1:0] →</td>
<td>DATATYPE[1:0] →</td>
</tr>
<tr>
<td>Swap management</td>
<td>Swap management</td>
</tr>
<tr>
<td>Decrypt</td>
<td>Decrypt</td>
</tr>
<tr>
<td>DATATYPE[1:0] ←</td>
<td>DATATYPE[1:0] ←</td>
</tr>
<tr>
<td>Swap management</td>
<td>Swap management</td>
</tr>
<tr>
<td>O1</td>
<td>O2</td>
</tr>
<tr>
<td>DOUT (plaintext P1)</td>
<td>DOUT (plaintext P2)</td>
</tr>
</tbody>
</table>

In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC, a special key scheduling is required before processing the first block.
**Figure 297** and **Figure 298** describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.

**Figure 297. CBC encryption**

**Figure 298. CBC decryption**

For more details, refer to NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.

**ECB and CBC encryption process**

This process is described in Section 33.4.5, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register as follows:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in encryption mode (write MODE[1:0] with 0x0).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the initialization vector into the AES_IVRx registers if CBC mode is selected in the previous step.

4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see Section 33.4.14).

5. Wait until KEYVALID is set (the key loading completed).

6. Enable the AES peripheral, by setting EN.

7. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in Section 33.4.7.
   b) Append the cleartext block into AES as described in Section 33.4.5, then read the AES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step b) until the third-last plaintext block is encrypted. For the last two blocks, follow the steps a) and b).

8. Finalize the sequence: disable the AES peripheral, by clearing EN.

**ECB/CBC decryption process**

This process is described in Section 33.4.5, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.

2. Initialize the AES_CR register as follows:
   - Select the key derivation mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.

3. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see Section 33.4.14).

4. Wait until KEYVALID is set (the key loading completed).

5. Enable the AES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.

6. Wait until the CCF flag in the AES_ISR register is set.

7. Clear the CCF flag, by setting the CCF bit of the AES_ICR register. The decryption key is available in the AES core and AES is disabled automatically.

8. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in decryption mode (write MODE[1:0] with 0x2). Do not change other parameters.

9. Write the initialization vector into the AES_IVRx registers if CBC mode is selected in the previous step.

10. Enable the AES peripheral, by setting EN.
11. Append encrypted data:
   a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in Section 33.4.7.
   b) Append the ciphertext block into AES as described in Section 33.4.5, then read the AES_DOUTR register four times to save the cleartext block (MSB first).
   c) Repeat the step b) until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps a) and b).
12. Finalize the sequence: disable the AES peripheral, by clearing EN.

**Suspend/resume operations in ECB/CBC modes**

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the input FIFO, by clearing the DMAINEN bit of the AES_CR register.
2. If DMA is not used, read four times the AES_DOUTR register to save the last processed block. If DMA is used, wait until the CCF flag is set in the AES_ISR register then stop the DMA transfers from the output FIFO, by clearing the DMAOUTEN bit of the AES_CR register.
3. If DMA is not used, wait until the CCF flag in the AES_ISR register is set (computation completed).
4. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
5. Save initialization vector registers (only required in CBC mode as the AES_IVRx registers are altered during the data processing).
6. Disable the AES peripheral, by clearing EN.
7. Save the AES_CR register and clear the key registers if they are not needed, to process the higher-priority message.
8. If DMA is used, save the DMA controller status (pointers for AES input and output data transfers, number of remaining bytes, and so on).

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller so as to complete the remaining input FIFO and output FIFO transfers.
2. Disable the AES peripheral, by clearing EN.
3. Restore the AES_CR register (with correct KEYSIZE) then restore the AES_KEYRx registers. If KMOD[1:0] is at 0x2, the key must be transferred again from the SAES peripheral (see Section 33.4.14).
4. Prepare the decryption key, as described in ECB/CBC decryption process (only required for ECB or CBC decryption).
5. Restore the IVRx registers, using the saved configuration (only required in CBC mode).
6. Enable the AES peripheral, by setting EN.
7. If DMA is used, enable AES DMA transfers, by setting DMAINEN and DMAOUTEN.

*Note: It is not required to save the key registers as the application knows the original key.*

**33.4.10 AES counter (CTR) mode**

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key
scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in Figure 299.

Figure 299. Message construction in CTR mode

The structure of this message is:

- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector (IV)**: a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

**CTR encryption and decryption**

*Figure 300* describes the counter (CTR) chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.
Initialization vectors in AES must be initialized as shown in Table 302.

### Table 302. Counter mode initialization vector definition

<table>
<thead>
<tr>
<th>AES_IVR3[31:0]</th>
<th>AES_IVR2[31:0]</th>
<th>AES_IVR1[31:0]</th>
<th>AES_IVR0[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>32-bit counter = 0x0001</td>
</tr>
</tbody>
</table>

### CTR encryption and decryption process

This process is described in Section 33.4.5, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the initialization vector into the AES_IVRx registers according to Table 302.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see Section 33.4.14).
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the AES peripheral, by setting EN.
7. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into AES as described in Section 33.4.5, then read the AES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step b) until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps a) and b). For the last block, discard
the bits that are not part of the message when the last block is smaller than 16 bytes.

8. Finalize the sequence: disable the AES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in Section 33.4.9: AES basic chaining modes (ECB, CBC).

Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.

33.4.11 AES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

**Figure 301. Message construction in GCM**

The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV)**: a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also known as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} – 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message** $P$ is both authenticated and encrypted as ciphertext $C$, with a known length $\text{Len}(P)$ that may be non-multiple of 16 bytes, and cannot exceed $2^{32} - 2$ 16-byte blocks.

- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in Table 304.

The GCM standard specifies that ciphertext $C$ has the same bit length as the plaintext $P$.

When a part of the message (AAD or $P$) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

For more details, refer to NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*.

*Figure 302* describes the GCM chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

**Figure 302. GCM authenticated encryption**

The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in Table 303.

---

1264/3119  RM0481 Rev 1
The last block of a GCM message contains the AAD header length and the payload length information, as shown in Table 304.

### Table 303. Initialization of IV registers in GCM mode

<table>
<thead>
<tr>
<th>AES_IVR3[31:0]</th>
<th>AES_IVR2[31:0]</th>
<th>AES_IVR1[31:0]</th>
<th>AES_IVR0[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>32-bit counter = 0x0002</td>
</tr>
</tbody>
</table>

GCM encryption and decryption process

This process is described in Section 33.4.6, with the following sequence of events:

**GCM initialize**
1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
   - Select the GCM initialization phase, by writing GCMPH[1:0] with 0x0.
3. Write the initialization vector in AES_IVRx registers according to Table 303.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see Section 33.4.14).
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
7. Wait until the CCF flag is set in the AES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
8. Clear the CCF flag by setting the CCF bit of the AES_ICR register.

**GCM header phase**
9. Initialize header phase:
   a) Select the GCM header phase, by writing 0x1 to GCMPH[1:0]. Do not change the other configurations written during GCM initialization.
   b) Enable the AES peripheral, by setting EN.

### Table 304. GCM last block definition

<table>
<thead>
<tr>
<th>Word order to AES_DINR</th>
<th>First word</th>
<th>Second word</th>
<th>Third word</th>
<th>Fourth word</th>
</tr>
</thead>
</table>
10. Append header data:
   a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into AES as described in Section 33.4.5.
   c) Repeat the step b) until the second-last AAD data block is processed. For the last block, follow the steps a) and b).

   **Note:** *This phase can be skipped if there is no AAD, that is, Len(A) = 0.*

   No data are read during header phase.

**GCM payload phase**

11. Initialize payload phase:
   a) Select the GCM payload phase, by writing GMCPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
   b) If the header phase is skipped, enable the AES peripheral by setting EN.

12. Append payload data:
   a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into AES as described in Section 33.4.5, then read the AES_DOUTR register four times to save the resulting block.
   c) Repeat the step b) until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps a) and b). For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

   **Note:** *This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).*

**GCM finalization**

13. Select the GCM final phase, by writing GMCPH[1:0] with 0x3. Do not change the other configurations written during GCM initialization.

14. Write the final GCM block into the AES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in Table 304.

15. Wait until the CCF flag in the AES_ISR register is set.

16. Get the GCM authentication tag, by reading the AES_DOUTR register four times.

17. Clear the CCF flag, by setting the CCF bit in AES_ICR register.

18. Disable the AES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

   **Note:** *In the final phase, data are written to AES_DINR normally (no swapping), while swapping is applied to tag data read from AES_DOUTR.*

   *When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.*
Suspend/resume operations in GCM mode

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the input FIFO, by clearing DMAINEN. If DMA is not used, wait until the CCF flag in the AES_ISR register is set (computation completed).
2. In the payload phase, if DMA is not used, read four times the AES_DOUTR register to save the last-processed block. If DMA is used, wait until the CCF flag in the AES_ISR register is set then stop the DMA transfers from the output FIFO, by clearing DMAOUTEN.
3. Clear the CCF flag of the AES_ISR register, by setting the CCF bit of the AES_ICR register.
4. Save the AES_SUSPRx registers in the memory.
5. In the payload phase, save the AES_IVRx registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
6. Disable the AES peripheral, by clearing EN.
7. Save the current AES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.
8. If DMA is used, save the DMA controller status (pointer for AES input data transfers, number of remaining bytes, and so on). In the payload phase, also save the pointer for AES output data transfers.

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the remaining input FIFO transfers. In the payload phase, also configure the DMA controller for the remaining output FIFO transfers.
2. Disable the AES peripheral, by clearing EN.
3. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPRx registers.
4. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers. In the header phase, write initial setting values back into the AES_IVRx registers.
5. Restore the initial setting values in the AES_CR and AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see Section 33.4.14).
6. Enable the AES peripheral, by setting EN.
7. If DMA is used, enable AES DMA requests, by setting DMAINEN (and DMAOUTEN if in payload phase).

33.4.12 AES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.
For more details, refer to NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.

Figure 304 describes the GMAC chaining implementation in the AES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

**Suspend/resume operations in GMAC**

In GMAC mode, the sequence described for the GCM applies except that only the header phase can be interrupted.
33.4.13 AES counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 305. Message construction in CCM mode

The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - Q: a bit string representation of the octet length of P (\(\text{Len}(P)\))
  - Nonce (N): a single-use value (that is, a new nonce must be assigned to each new communication) of \(\text{Len}(N)\) size. The sum \(\text{Len}(N) + \text{Len}(P)\) must be equal to 15 bytes.
  - Flags: most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values \(t\) (MAC length expressed in bytes) and \(Q\) (plaintext length such that \(\text{Len}(P) < 2^{8Q}\) bytes). The counter blocks range associated to \(Q\) is equal to \(2^{8Q-4}\), that is, if the maximum value of \(Q\) is 8, the counter blocks used in cipher must be on 60 bits.

- **16-byte blocks (B)** associated to the associated data (A). This part of the message is only authenticated, not encrypted. This section has a known length \(\text{Len}(A)\) that can be a non-multiple of 16 bytes (see Figure 305). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:
  - If \(0 < a < 2^{16} - 2^5\), then it is encoded as \([a]_{16}\), that is, on two bytes.
  - If \(2^{16} - 2^8 < a < 2^{32}\), then it is encoded as \(0xff \| 0xfe \| [a]_{32}\), that is, on six bytes.
  - If \(2^{32} < a < 2^{64}\), then it is encoded as \(0xff \| 0xff \| [a]_{64}\), that is, on ten bytes.
• **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see Figure 305).

• **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

*Note:* CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

- N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)
- A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)
- P: 20212223 (Len(P) = 32 bits or 4 bytes)
- T: 6084341B (Len(T) = 32 bits or t = 4)
- B0: 4F101112 13141516 00000000 00000004
- B1: 00080001 02030405 06070000 00000000
- B2: 20212223 00000000 00000000 00000000
- CTR0: 0710111213 141516 00000000 00000000
- CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality.
**Figure 306** describes the CCM chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

The first block of a CCM message (B0) must be prepared by the application as defined in Table 305.

### Table 305. Initialization of IV registers in CCM mode

<table>
<thead>
<tr>
<th>AES_IVR3[31:0]</th>
<th>AES_IVR2[31:0]</th>
<th>AES_IVR1[31:0]</th>
<th>AES_IVR0[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0[127:96]</td>
<td>B0[95:64]</td>
<td>B0[63:32]</td>
<td>B0[31:0]</td>
</tr>
</tbody>
</table>

1. The 5 most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

AES supports counters up to 64 bits, as specified by NIST.
CCM encryption and decryption process

This process is described in Section 33.4.6, with the following sequence of events:

CCM initialize
1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
   - Select the CCM initialization phase, by writing GCMPH[1:0] with 0x0.
3. Write the B0 data in AES_IVRx registers according to Table 305.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see Section 33.4.14).
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
7. Wait until the CCF flag in the AES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.

CCM header phase
9. Initialize header phase:
   a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
   b) Select the CCM header phase, by writing GCMPH[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
   c) Enable the AES peripheral, by setting EN.
10. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in Section 33.4.5.
    c) Repeat the step b) until the second-last associated data block is processed. For the last block, follow the steps a) and b).

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0
No data are read during the header phase.

CCM payload phase
11. Initialize payload phase:
    a) Select the CCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
    b) If the header phase is skipped, enable the AES peripheral, by setting EN.
12. Append payload data:
   a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into AES as described in Section 33.4.5, then read the AES_DOUTR register four times to save the resulting block.
   c) Repeat the step b) until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps a) and b). For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

   Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove LSB_{Len(T)} encrypted tag information when decrypting ciphertext C.

CCM finalization

13. Select the CCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.

14. Wait until CCF flag in the AES_ISR register is set.

15. Get the CCM authentication tag, by reading the AES_DOUTR register four times.

16. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.

17. Disable the AES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

   Note: In the final phase, swapping is applied to tag data read from AES_DOUTR register.

When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

To suspend the processing of a message in header or payload phase, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the input FIFO, by clearing DMAINEN. If DMA is not used, wait until the CCF flag of the AES_ISR register is set (computation completed).

2. In the payload phase, if DMA is not used, read four times the AES_DOUTR register to save the last-processed block. If DMA is used, wait until the CCF flag in the AES_ISR register is set, then stop the DMA transfers from the output FIFO, by clearing DMAOUTEN.

3. Clear the CCF flag in the AES_ISR register, by setting the CCF bit of the AES_ICR register.

4. Save the AES_SUSPRx registers in the memory.

5. Save the IV registers as they are altered during the data processing.

6. Disable the AES peripheral, by clearing EN.

7. Save the current AES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

8. If DMA is used, save the DMA controller status (pointer for AES input data transfers, number of remaining bytes, and so on). In the payload phase, also save pointer for AES output data transfers.
To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the remaining input FIFO transfers. In the payload phase, also configure the DMA controller for the remaining output FIFO transfers.
2. Disable the AES peripheral, by clearing EN.
3. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPRx registers.
4. Restore AES_IVRx registers using the saved configuration.
5. Restore the initial setting values in the AES_CR and AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key must be transferred again from the SAES peripheral (see Section 33.4.14).
6. Enable the AES peripheral, by setting EN.
7. If DMA is used, enable AES DMA requests, by setting DMAINEN (and DMAOUTEN if in payload phase).

33.4.14 AES key sharing with secure AES co-processor

The AES peripheral can use the SAES peripheral as security co-processor. The secure application prepares the key in the robust SAES peripheral and when it is ready, the AES application can load this prepared key through a dedicated hardware key bus.

The recommended sequence is described hereafter and in the section SAES operations with shared keys in the SAES section of this document.

1. In SAES peripheral, the application encrypts (wraps) the key to share in Shared-key mode (KMOD[1:0] at 0x2).
2. Each time the shared key is required in AES peripheral, the application decrypts it in the SAES peripheral in Shared-key mode (KMOD[1:0] at 0x2).
3. Once the shared key is decrypted (unwrapped) and loaded in SAES_KEYRx registers it can be shared with AES. To load the shared key in AES, the application sets KEYSIZE as appropriate and writes KMOD[1:0] with 0x2. When KEYVALID is cleared, the key is automatically transferred by hardware into the AES_KEYRx registers and the BUSY flag in the AES_SR register set.
4. Once the key transfer is completed, the BUSY flag is cleared and the KEYVALID flag set in the AES_SR register. If KEYVALID is not set when BUSY bit is cleared, or if the KEIF flag is set in the AES_ISR register, either the KEYSIZE value is incorrect or an unexpected event occurred during the transfer (such as DPA error, tamper event or KEYVALID cleared before the end of the transfer). When such errors occur, reset both peripherals through their IPRST bits and restart the whole key sharing process.

When the key sharing sequence is completed, the AES is initialized with a valid, shared key. The application can then process data in normal key mode, by writing KMOD[1:0] with 0x0.

Note: This sequence in AES peripheral can be run multiple times (for example, to manage a suspend/resume situation), as long as SAES peripheral is unused and duly remains in key sharing state.
33.4.15 AES data registers and data swapping

Data input and output

A 16-byte data block enters the AES peripheral with four successive 32-bit word writes into the AES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the AES peripheral with four successive 32-bit word reads of the AES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See Table 306 "no swapping" option for details.

Data swapping

The AES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the AES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the AES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through DATATYPE[1:0]. The selection applies to both AES input and output.

Note: The data in AES key registers (AES_KEYRx) and initialization vector registers (AES_IVRx) are not sensitive to the swap mode selection.

The AES data swapping feature is summarized in Table 306 and Figure 307.

Table 306. AES data swapping example

<table>
<thead>
<tr>
<th>DATATYPE[1:0]</th>
<th>Swapping performed</th>
<th>Data block</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>System memory data (big-endian)</td>
</tr>
</tbody>
</table>
| 0x0           | No swapping        | Block[127..64]: 0x04E6F772 2E04CE96  
|               |                    | Block[63..0]: 0x4E6F7720 69732074  |
| 0x1           | Half-word (16-bit) swapping | Block[63..0]: 0x4E6F 7720 6973 2074  |
|               |                    | Address @, word[127..96]: 0x04E6F772  
|               |                    | Address @ + 0x4, word[95..64]: 0x2E04CE96  
|               |                    | Address @ + 0x8, word[63..32]: 0x4E6F7720  
|               |                    | Address @ + 0xC, word[31..0]: 0x69732074  |
| 0x2           | Byte (8-bit) swapping | Block[63..0]: 0x4E 6F 77 20 69 73 20 74  |
|               |                    | Address @, word[63..32]: 0x7720 4E6F  
|               |                    | Address @ + 0x4, word[31..0]: 0x2074 6973  |
Table 306. AES data swapping example (continued)

<table>
<thead>
<tr>
<th>DATATYPE[1:0]</th>
<th>Swapping performed</th>
<th>Data block</th>
<th>System memory data (big-endian)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x3</td>
<td>Bit swapping</td>
<td>Block[63..32]: 0x4E6F7720</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0100  1110  0110  1111  0111  0110  0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Block[31..0]: 0x69732074</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0110  1001  0111  0011  0010  0000  0111  0100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @, word[63..32]: 0x04EE F672</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0000  0100  1110  1110  1111  0110  0111  0010</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0x4, word[31..0]: 0x2E04 CE96</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0010  1110  0000  0100  1110  1110  1001  0110</td>
<td></td>
</tr>
</tbody>
</table>

Figure 307. 128-bit block construction according to the data type

**DATATYPE[1:0] = 00: no swapping**

**DATATYPE[1:0] = 01: 16-bit (half-word) swapping**

**DATATYPE[1:0] = 10: 8-bit (byte) swapping**

**DATATYPE[1:0] = 11: bit swapping**

Legend:  
- **Dx**: input/output data bit ‘x’  
- **↓**: Data swap  
- **↓↓**: Order of write to data input / read from data output register  
- **----**: Zero padding (example)  
- __AES core input/output data__
Data padding

*Figure 307* also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

### 33.4.16 AES key registers

The eight AES_KEYRx write-only registers store the encryption or decryption key information, as shown on Table 307. Reads are not allowed for security reason.

*Note:* In memory and in AES key registers, keys are stored in little-endian format, with most significant byte on the highest address.

*Table 307. Key endianness in AES_KEYRx registers (128/256-bit keys)*

<table>
<thead>
<tr>
<th>AES_KEYR7 [31:0]</th>
<th>AES_KEYR6 [31:0]</th>
<th>AES_KEYR5 [31:0]</th>
<th>AES_KEYR4 [31:0]</th>
<th>AES_KEYR3 [31:0]</th>
<th>AES_KEYR2 [31:0]</th>
<th>AES_KEYR1 [31:0]</th>
<th>AES_KEYR0 [31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>KEY[127:96]</td>
<td>KEY[95:64]</td>
<td>KEY[63:32]</td>
<td>KEY[31:0]</td>
</tr>
</tbody>
</table>

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the AES_KEYRx registers are ignored when AES peripheral is enabled (EN bit set). The application must check this before modifying key registers.

The entire key must be written before starting an AES computation. In normal key mode (KMOD[1:0] at 0x0), the key registers must always be written in either ascending or descending order. The write sequence becomes:

- AES_KEYRx (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- AES_KEYRx (x = 0 to 7 or x=7 to 0) for KEYSIZE set

*Note:* KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see *Section 33.4.18*).

### 33.4.17 AES initialization vector registers

The four AES_IVRx registers store the initialization vector (IV) information, as shown in Table 308. They can only be written if the AES peripheral is disabled (EN cleared).

*Note:* In memory and in AES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.
Initialization vector information depends on the chaining mode selected. When used, AES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

33.4.18 AES error management

The AES peripheral manages the errors described in this section.

Read error flag (RDERRF)

Unexpected read attempt of the AES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: **AES is not disabled when RDERRF rises and it continues processing.**

An interrupt is generated if the RWEIE bit is set. For more details, refer to Section 33.5: AES interrupts.

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the AES_ICR register.

Write error flag (WDERR)

Unexpected write attempt of the AES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

Note: **AES is not disabled when WRERRF rises and it continues processing.**

An interrupt is generated if the RWEIE bit is set. For more details, refer to Section 33.5: AES interrupts.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the AES_ICR register.

Key error interrupt flag (KEIF)

There are multiple sources of errors that set the KEIF flag of the AES_ISR register and clear the KEYVALID bit of the AES_SR register:

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See Section 33.4.16: AES key registers for details.
- **Key sharing size mismatch error:** triggered when KMOD[1:0] is at 0x2 and KEYSIZE in AES peripheral does not match KEYSIZE in SAES peripheral.

<table>
<thead>
<tr>
<th>AES_IVR3[31:0]</th>
<th>AES_IVR2[31:0]</th>
<th>AES_IVR1[31:0]</th>
<th>AES_IVR0[31:0]</th>
</tr>
</thead>
</table>

Table 308. IVI bitfield spread over AES_IVRx registers
• **Key sharing error**: triggered upon failing transfer of SAES shared key to AES peripheral. See Section 33.4.14: AES key sharing with secure AES co-processor for details.

The KEIF flag is cleared with corresponding bit of the AES_ICR register. An interrupt is generated if the KEIE bit of the AES_IER register is set. For more details, refer to Section 33.5: AES interrupts.

Upon a key sharing error, reset both AES and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

*Note:* *For any key error, clear KEIF flag prior to disabling and re-configuring AES.*

### 33.5 AES interrupts

There are multiple individual maskable interrupt sources generated by the AES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)

See Section 33.4.18: AES error management for details on AES errors.

These sources are combined into a common interrupt signal from the AES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable AES interrupt sources individually by setting/clearing the corresponding enable bit of the AES_IER register.

The status of the individual maskable interrupt sources can be read from the AES_ISR register. They are cleared by setting the corresponding bit of the AES_ICR register.

*Table 309* gives a summary of the available features.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES</td>
<td>computation completed flag</td>
<td>CCF</td>
<td>CCFIE</td>
<td>set CCF(^{(1)})</td>
</tr>
<tr>
<td></td>
<td>read error flag</td>
<td>RDERRF(^{(2)})</td>
<td>RWEIE</td>
<td>set RWEIF(^{(1)})</td>
</tr>
<tr>
<td></td>
<td>write error flag</td>
<td>WRERRF(^{(2)})</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>key error flag</td>
<td>KEIF</td>
<td>KEIE</td>
<td>set KEIF(^{(1)})</td>
</tr>
</tbody>
</table>

1. Bit of the AES_ICR register.
2. Flag of the AES_SR register, mirrored by the flag RWEIF of the AES_ISR register.

### 33.6 AES DMA requests

The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the AES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.
AES only supports single DMA requests.

Detailed usage of DMA with AES can be found in **Appending data using DMA** subsection of **Section 33.4.5: AES encryption or decryption typical usage**.

**Data input using DMA**

Setting DMAINEN enables DMA writing into AES. AES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the AES_DINR register (quadruple 32-bit word, MSB first).

*Note:* According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see **Section 33.4.7**).

**Data output using DMA**

Setting DMAOUTEN enables DMA reading from AES. AES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the AES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, AES switches automatically to a new input phase for the next data block, if any.

In DMA mode, the CCF flag has no use because the reading of the AES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to managing the data transfers by software.

*Note:* According to the message size, extra bytes might need to be discarded by application in the last block.

**Stopping DMA transfers**

All DMA request signals are de-asserted when AES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

### 33.7 AES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

**Table 310. Processing latency for ECB, CBC and CTR**

<table>
<thead>
<tr>
<th>Key size</th>
<th>Mode of operation</th>
<th>Chaining algorithm</th>
<th>Clock cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>128-bit</td>
<td>Encryption or decryption(1)</td>
<td>ECB, CBC, CTR</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>Key preparation</td>
<td>-</td>
<td>59</td>
</tr>
<tr>
<td>256-bit</td>
<td>Encryption or decryption(1)</td>
<td>ECB, CBC, CTR</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>Key preparation</td>
<td>-</td>
<td>82</td>
</tr>
</tbody>
</table>

1. Excluding key preparation time (ECB and CBC only).

**Table 311. Processing latency for GCM and CCM (in clock cycles)**

<table>
<thead>
<tr>
<th>Key size</th>
<th>Mode of operation</th>
<th>Chaining algorithm</th>
<th>Initialization phase</th>
<th>Header phase(1)</th>
<th>Payload phase(1)</th>
<th>Final phase(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128-bit</td>
<td>Encryption/ Decryption</td>
<td>GCM</td>
<td>64</td>
<td>35</td>
<td>51</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCM</td>
<td>63</td>
<td>55</td>
<td>114</td>
<td>58</td>
</tr>
</tbody>
</table>
Table 311. Processing latency for GCM and CCM (in clock cycles) (continued)

<table>
<thead>
<tr>
<th>Key size</th>
<th>Mode of operation</th>
<th>Chaining algorithm</th>
<th>Initialization phase</th>
<th>Header phase&lt;sup&gt;(1)&lt;/sup&gt;</th>
<th>Payload phase&lt;sup&gt;(1)&lt;/sup&gt;</th>
<th>Final phase&lt;sup&gt;(1)&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>256-bit</td>
<td>Encryption/Decryption</td>
<td>GCM</td>
<td>88</td>
<td>35</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCM</td>
<td>87</td>
<td>79</td>
<td>162</td>
<td>82</td>
</tr>
</tbody>
</table>

1. Data insertion can include wait states forced by AES on the AHB bus (maximum 3 cycles, typical 1 cycle).
33.8 AES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

33.8.1 AES control register (AES_CR)

Address offset: 0x000
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>IPRST: AES peripheral software reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Setting the bit resets the AES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data are lost. For this reason, it is recommended to set the bit before handing over the AES to a less secure application. The bit must be kept low while writing any configuration registers.</td>
</tr>
<tr>
<td>Bits 30:26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 25:24</td>
<td>KMOD[1:0]: Key mode selection</td>
</tr>
<tr>
<td></td>
<td>The bitfield defines how the AES key can be used by the application. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.</td>
</tr>
<tr>
<td></td>
<td>0x0: Normal key mode. Key registers are freely usable.</td>
</tr>
<tr>
<td></td>
<td>0x2: Shared key mode. If shared key mode is properly initialized in SAES peripheral, the AES peripheral automatically loads its key registers with the data stored in the SAES key registers. The key value is available in AES key registers when BUSY bit is cleared and KEYVALID is set in the AES_SR register. Key error flag KEIF is set otherwise in the AES_ISR register. Others: Reserved</td>
</tr>
<tr>
<td></td>
<td>Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.</td>
</tr>
<tr>
<td>Bits 23:20</td>
<td>NPBLB[3:0]: Number of padding bytes in last block</td>
</tr>
<tr>
<td></td>
<td>This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.</td>
</tr>
<tr>
<td></td>
<td>0x0: All bytes are valid (no padding)</td>
</tr>
<tr>
<td></td>
<td>0x1: Padding for the last LSB byte</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>0xFF: Padding for the 15 LSB bytes of last block.</td>
</tr>
<tr>
<td>Bit 19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bit 18 **KEYSIZE**: Key size selection
- This bitfield defines the key length in bits of the key used by AES.
  0: 128-bit
  1: 256-bit
  Attempts to write the bit are ignored when BUSY is set, as well as when the EN is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **GCMPTH[1:0]**: GCM or CCM phase selection
- This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.
  0x0: Initialization phase
  0x1: Header phase
  0x2: Payload phase
  0x3: Final phase

Bit 12 **DMAOUTEN**: DMA output enable
- This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from AES via DMA.
  0: Disable
  1: Enable
  Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 **DMAINEN**: DMA input enable
- This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to AES via DMA.
  0: Disable
  1: Enable
  Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7 Reserved, must be kept at reset value.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode
- This bitfield selects the AES chaining mode:
  0x0: Electronic codebook (ECB)
  0x1: Cipher-block chaining (CBC)
  0x2: Counter mode (CTR)
  0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)
  0x4: Counter with CBC-MAC (CCM)
  others: Reserved
  Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: Operating mode
- This bitfield selects the AES operating mode:
  0x0: Encryption
  0x1: Key derivation (or key preparation), for ECB/CBC decryption only
  0x2: Decryption
  0x3: Reserved
  Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.
33.8.2 AES status register (AES_SR)

Address offset: 0x004
Reset value: 0x0000 0000

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

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

Bit 7 KEYVALID: Key valid flag
This bit is set by hardware when the key of size defined by KEYSIZE is loaded in AES_KEYRx key registers.
0: Key not valid
1: Key valid
The EN bit can only be set when KEYVALID is set.
In normal mode when KMOD[1:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.
When KMOD[1:0] is different from zero, the BUSY flag is automatically set by AES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is usually set, BUSY cleared and KEYVALID remains cleared.
If set, KEIF must be cleared through the AES_ICR register, otherwise KEYVALID cannot be set. See the KEIF flag description for more details.
For further information on key loading, refer to Section 33.4.16: AES key registers.

Bits 6:4 Reserved, must be kept at reset value.
Bit 3  **BUSY**: Busy  
This flag indicates whether AES is idle or busy.  
0: Idle  
1: Busy  
AES is flagged as idle when disabled (when EN is low) or when the last processing is completed.  
AES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), or transferring a shared key from the SAES peripheral.  
When GCM encryption is selected, this flag must be at zero before suspending current process to manage a higher-priority message.

Bit 2  **WRERRF**: Write error flag  
This bit is set when an unexpected write to the AES_DINR register occurred. When set WRERRF bit has no impact on the AES operations.  
0: No error  
1: Unexpected write to AES_DINR register occurred during computation or data output phase.  
The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.  
The flag is cleared by setting the RWEIF bit of the AES_ICR register.

Bit 1  **RDERRF**: Read error flag  
This bit is set when an unexpected read to the AES_DOUTR register occurred. When set RDERRF bit has no impact on the AES operations.  
0: No error  
1: Unexpected read to AES_DOUTR register occurred during computation or data input phase.  
The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.  
The flag is cleared by setting the RWEIF bit of the AES_ICR register.

Bit 0  Reserved, must be kept at reset value.

### 33.8.3  AES data input register (AES_DINR)

Address offset: 0x008  
Reset value: 0x0000 0000

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

**DIN[31:16]**

**DIN[15:0]**

Bits 31:0  **DIN[31:0]**: Data input  
A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes block of input data to the AES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 16-bytes input buffer.  
Reads return zero.
33.8.4 AES data output register (AES_DOUTR)

Address offset: 0x00C
Reset value: 0x0000 0000

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

**DOUT[31:16]:**

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

Bits 31:0 **DOUT[31:0]:** Data output
This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte block of output data from the AES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

33.8.5 AES key register 0 (AES_KEYR0)

Address offset: 0x010
Reset value: 0x0000 0000

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

**KEY[31:16]:**

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

**KEY[15:0]:**

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

Bits 31:0 **KEY[31:0]:** Cryptographic key, bits [31:0]
These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the AES_CR register.

Writes to AES_KEYRx registers are ignored when AES is enabled (EN bit set). When the key comes from the SAES peripheral (KMOD[1:0] at 0x2), writes to key registers are also ignored and they result in setting the KEIF bit of the AES_ISR register.

With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.
33.8.6 AES key register 1 (AES_KEYR1)

Address offset: 0x014
Reset value: 0x0000 0000

Bits 31:0 KEY[63:32]: Cryptographic key, bits [63:32]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

33.8.7 AES key register 2 (AES_KEYR2)

Address offset: 0x018
Reset value: 0x0000 0000

Bits 31:0 KEY[95:64]: Cryptographic key, bits [95:64]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

33.8.8 AES key register 3 (AES_KEYR3)

Address offset: 0x01C
Reset value: 0x0000 0000

Bits 31:0 KEY[127:96]: Cryptographic key, bits [127:96]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.
### 33.8.9 AES initialization vector register 0 (AES_IVR0)

Address offset: 0x020  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVI[31:16]</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]

AES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable). Write to this register is ignored when EN bit is set in AES_SR register.

### 33.8.10 AES initialization vector register 1 (AES_IVR1)

Address offset: 0x024  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 63:32</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVI[63:32]</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]

Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

### 33.8.11 AES initialization vector register 2 (AES_IVR2)

Address offset: 0x028  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 95:64</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVI[95:64]</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]

Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.
33.8.12 AES initialization vector register 3 (AES_IVR3)
Address offset: 0x02C
Reset value: 0x0000 0000

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

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]
Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

33.8.13 AES key register 4 (AES_KEYR4)
Address offset: 0x030
Reset value: 0x0000 0000

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

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

33.8.14 AES key register 5 (AES_KEYR5)
Address offset: 0x034
Reset value: 0x0000 0000

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

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.
33.8.15 AES key register 6 (AES_KEYR6)
Address offset: 0x038
Reset value: 0x0000 0000

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

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

33.8.16 AES key register 7 (AES_KEYR7)
Address offset: 0x03C
Reset value: 0x0000 0000

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

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

33.8.17 AES suspend registers (AES_SUSPRx)
Address offset: 0x040 + x * 0x4, (x = 0 to 7)
Reset value: 0x0000 0000

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

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

1290/3119    RM0481 Rev 1
Bits 31:0 **SUSP[31:0]:** Suspend data

AES_SUSPRx registers contain the complete internal register states of the AES when the GCM, GMAC or CCM processing of the current task is suspended to process a higher-priority task. Refer to Section 33.4.8: AES suspend and resume operations for more details.

Read to this register returns zero when EN bit is cleared in AES_SR register.

AES_SUSPRx registers are not used in other chaining modes than GCM, GMAC or CCM.

### 33.8.18 AES interrupt enable register (AES_IER)

Address offset: 0x300

Reset value: 0x0000 0000

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

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

Bit 2 **KEIE:** Key error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when KEIF (key error flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

Bit 1 **RWEIE:** Read or write error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when RWEIF (read and/or write error flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

Bit 0 **CCFIE:** Computation complete flag interrupt enable

This bit enables or disables (masks) the AES interrupt generation when CCF (computation complete flag) is set.
0: Disabled (masked)
1: Enabled (not masked)
33.8.19 AES interrupt status register (AES_ISR)

Address offset: 0x304
Reset value: 0x0000 0000

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

|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

| Bit 31:3 Reserved, must be kept at reset value. |
| Bit 2 KEIF: Key error interrupt flag |
| This read-only bit is set by hardware when the key information fails to load into key registers. |
| 0: No key error detected |
| 1: Key information failed to load into key registers |
| The flag setting generates an interrupt if the KEIE bit of the AES_IER register is set. |
| The flag is cleared by setting the corresponding bit of the AES_ICR register. |
| KEIF is raised upon any of the following events: |
| –AES_KEYRx register write does not respect the correct order. (For KEYSIZE cleared, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 register, or reverse. For KEYSIZE set, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 then AES_KEYR4 then AES_KEYR5 then AES_KEYR6 then AES_KEYR7, or reverse). |
| –AES fails to load the key shared by the SAES peripheral (KMOD[1:0] = 0x2). |
| KEIF must be cleared by the application software, otherwise KEYVALID cannot be set. |

| Bit 1 RWEIF: Read or write error interrupt flag |
| This read-only bit is set by hardware when a RDERRF or a WRERRF error flag is set in the AES_SR register. |
| 0: No read or write error detected |
| 1: Read or write error detected |
| The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set. |
| The flag is cleared by setting the corresponding bit of the AES_ICR register. |
| The flags has no meaning when key derivation mode is selected. |
| See the AES_SR register for details. |

| Bit 0 CCF: Computation complete flag |
| This flag indicates whether the computation is completed. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set. |
| 0: Not completed |
| 1: Completed |
| The flag setting generates an interrupt if the CCFIE bit of the AES_IER register is set. |
| The flag is cleared by setting the corresponding bit of the AES_ICR register. |
33.8.20  AES interrupt clear register (AES_ICR)

Address offset: 0x308
Reset value: 0x0000 0000

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

Bit 2  KEIF: Key error interrupt flag clear
   Setting this bit clears the KEIF status bit of the AES_ISR register.

Bit 1  RWEIF: Read or write error interrupt flag clear
   Setting this bit clears the RWEIF status bit of the AES_ISR register, and clears both RDERRF and
   WRERRF flags in the AES_SR register.

Bit 0  CCF: Computation complete flag clear
   Setting this bit clears the CCF status bit of the AES_ISR register.

33.8.21  AES register map

Table 312. AES register map and reset values

| Offset  | Register name | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|---------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x000   | AES_CR        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x004   | AES_SR        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x008   | AES_DINR      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x00C   | AES_DOUTR     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x010   | AES_KEYR0     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x014   | AES_KEYR1     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
### Table 312. AES register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register offset (hex)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x018</td>
<td>AES_KEYR2</td>
<td>0x00000000000000000000000000000000</td>
<td>KEY[95:64]</td>
</tr>
<tr>
<td></td>
<td>AES_KEYR3</td>
<td>0x00000000000000000000000000000000</td>
<td>KEY[127:96]</td>
</tr>
<tr>
<td>0x020</td>
<td>AES_IVR0</td>
<td>0x00000000000000000000000000000000</td>
<td>IV[31:0]</td>
</tr>
<tr>
<td>0x024</td>
<td>AES_IVR1</td>
<td>0x00000000000000000000000000000000</td>
<td>IV[63:32]</td>
</tr>
<tr>
<td>0x028</td>
<td>AES_IVR2</td>
<td>0x00000000000000000000000000000000</td>
<td>IV[95:64]</td>
</tr>
<tr>
<td>0x02C</td>
<td>AES_IVR3</td>
<td>0x00000000000000000000000000000000</td>
<td>IV[127:96]</td>
</tr>
<tr>
<td>0x030</td>
<td>AES_KEYR4</td>
<td>0x00000000000000000000000000000000</td>
<td>KEY[159:128]</td>
</tr>
<tr>
<td>0x034</td>
<td>AES_KEYR5</td>
<td>0x00000000000000000000000000000000</td>
<td>KEY[191:160]</td>
</tr>
<tr>
<td>0x038</td>
<td>AES_KEYR6</td>
<td>0x00000000000000000000000000000000</td>
<td>KEY[223:192]</td>
</tr>
<tr>
<td>0x03C</td>
<td>AES_KEYR7</td>
<td>0x00000000000000000000000000000000</td>
<td>KEY[255:224]</td>
</tr>
<tr>
<td>0x040</td>
<td>AES_SUSPR0</td>
<td>0x00000000000000000000000000000000</td>
<td>SUSP[31:0]</td>
</tr>
<tr>
<td>0x044</td>
<td>AES_SUSPR1</td>
<td>0x00000000000000000000000000000000</td>
<td>SUSP[31:0]</td>
</tr>
<tr>
<td>0x048</td>
<td>AES_SUSPR2</td>
<td>0x00000000000000000000000000000000</td>
<td>SUSP[31:0]</td>
</tr>
<tr>
<td>0x04C</td>
<td>AES_SUSPR3</td>
<td>0x00000000000000000000000000000000</td>
<td>SUSP[31:0]</td>
</tr>
<tr>
<td>0x050</td>
<td>AES_SUSPR4</td>
<td>0x00000000000000000000000000000000</td>
<td>SUSP[31:0]</td>
</tr>
<tr>
<td>0x054</td>
<td>AES_SUSPR5</td>
<td>0x00000000000000000000000000000000</td>
<td>SUSP[31:0]</td>
</tr>
<tr>
<td>0x058</td>
<td>AES_SUSPR6</td>
<td>0x00000000000000000000000000000000</td>
<td>SUSP[31:0]</td>
</tr>
<tr>
<td>0x05C</td>
<td>AES_SUSPR7</td>
<td>0x00000000000000000000000000000000</td>
<td>SUSP[31:0]</td>
</tr>
<tr>
<td>0x060-</td>
<td>Reserved</td>
<td>0x00000000000000000000000000000000</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x2FF</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Refer to Section 2.3 on page 111 for the register boundary addresses.
34  Secure AES coprocessor (SAES)

34.1 Introduction

The secure AES coprocessor (SAES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST. It incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3.

SAES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128 or 256 bits, as well as special modes such as hardware secret key encryption/decryption (Wrapped-key mode) and key sharing with faster AES peripheral (Shared-key mode).

SAES has the possibility to load by hardware STM32 hardware secret master keys (boot hardware key BHK and derived hardware unique key DHUK), usable but not readable by application.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required). It is hardware-linked with the true random number generator (TRNG) and with the AES peripheral.
34.2 SAES main features

- Compliant with NIST FIPS publication 197 “Advanced encryption standard (AES)” (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- Protection against side-channel attacks (SCA), incl. differential power analysis (DPA), certified SESIP and PSA security assurance level 3
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 480 or 680 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Hardware secret key encryption/decryption (Wrapped-key mode)
- Using dedicated key bus, optional key sharing with faster AES peripheral (Shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
  - Optional 128-bit or 256-bit hardware loading of two hardware secret keys (BHK, DHUK) that can be XOR-ed together
- Security context enforcement for keys
- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.
- Possibility for software to suspend a message if SAES needs to process another message with a higher priority, then resume the original message

34.3 SAES implementation

The devices have one SAES peripheral, implemented as per the following table. It shares the key with the AES peripheral. For comparison, the AES peripheral is also included in the table.
Table 313. AES versus SAES features

<table>
<thead>
<tr>
<th>Modes or features(1)</th>
<th>AES</th>
<th>SAES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECB, CBC chaining</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CTR, CCM, GCM chaining</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>AES 128-bit ECB encryption in cycles</td>
<td>51</td>
<td>480</td>
</tr>
<tr>
<td>DHUK and BHK key selection</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Resistance to side-channel attacks</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Shared key between SAES and AES</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Key sizes in bits</td>
<td>128, 256</td>
<td>128, 256</td>
</tr>
</tbody>
</table>

1. X = supported.

34.4 SAES functional description

34.4.1 SAES block diagram

*Figure 308* shows the block diagram of SAES.

![SAES Block Diagram](figure308.png)

**Figure 308. SAES block diagram**

Note: AES represents the AES peripheral.

34.4.2 SAES internal signals

*Table 314* describes the user relevant internal signals interfacing the SAES peripheral.
### RM0481

**Secure AES coprocessor (SAES)**

#### 34.4.3 SAES reset and clocks

The SAES peripheral is clocked by the AHB bus clock. The SAES instance has also a dedicated kernel clock programmed in the RCC.

The SAES has a dedicated reset bit in the RCC.

#### 34.4.4 SAES symmetric cipher implementation

The secure AES coprocessor (SAES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in Table 315. Those functions can be certified NIST PUB 140-3.

##### Table 314. SAES internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>saes_hclk</td>
<td>Input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>saes_ker_ck</td>
<td>Input</td>
<td>SAES kernel clock</td>
</tr>
<tr>
<td>saes_it</td>
<td>Output</td>
<td>SAES interrupt request</td>
</tr>
<tr>
<td>saes_in_dma</td>
<td>Input/Output</td>
<td>SAES incoming data DMA single request/acknowledge</td>
</tr>
<tr>
<td>saes_out_dma</td>
<td>Input/Output</td>
<td>SAES processed data DMA single request/acknowledge</td>
</tr>
<tr>
<td>saes_itamp_out</td>
<td>Output</td>
<td>Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, SAES automatically clears key registers. A reset is required for SAES to be usable again.</td>
</tr>
<tr>
<td>RHUK</td>
<td>Input</td>
<td>256-bit root hardware unique key (non-volatile, unique per device and secret to software), used to internally compute the derived hardware unique key (DHUK)</td>
</tr>
<tr>
<td>BHK(^{(1)})</td>
<td>Input</td>
<td>256-bit boot hardware key (BHK) stored in tamper-resistant secure backup registers and written by a secure code during boot. Once written, this key cannot be read nor written by any application until the next product reset.</td>
</tr>
<tr>
<td>SOID</td>
<td>Input</td>
<td>Static operating ID hardware input used during each DHUK computation. Its change causes the erasure of key registers and sets the KEIF flag.</td>
</tr>
</tbody>
</table>

1. Connected to a set of backup registers in TAMP peripheral that are written, then read/write locked, by the application software (see Section 34.4.17 for details).

##### Table 315. SAES approved symmetric key functions

<table>
<thead>
<tr>
<th>Operations</th>
<th>Algorithm</th>
<th>Specification</th>
<th>Key bit lengths</th>
<th>Chaining modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption, decryption</td>
<td>AES</td>
<td>FIPS PUB 197 NIST SP800-38A, NIST SP800-38C</td>
<td>128, 256</td>
<td>ECB, CBC, CTR</td>
</tr>
<tr>
<td>Authenticated encryption or decryption</td>
<td>AES</td>
<td>NIST SP800-38C NIST SP800-38D</td>
<td></td>
<td>GCM, CCM</td>
</tr>
<tr>
<td>Cipher-based message authentication code</td>
<td>AES</td>
<td>NIST SP800-38D</td>
<td></td>
<td>GMAC</td>
</tr>
</tbody>
</table>
SAES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext).

It is possible to suspend then resume any SAES processing, following the sequence described in Section 34.4.8.

### 34.4.5 SAES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

**Figure 309. Encryption/decryption typical usage**

![SAES Encryption/Decryption Diagram](diagram)

**Initialization**

The SAES peripheral is initialized according to the chaining mode. Refer to Section 34.4.9: SAES basic chaining modes (ECB, CBC) and Section 34.4.10: SAES counter (CTR) mode for details.

**Data append**

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to Section 34.4.7: SAES ciphertext stealing and data padding if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.
Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. When KEYVALID is set, enable the SAES peripheral, by setting the EN bit of the SAES_CR register (if not already done).
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the SAES_DINR register.
   b) Wait until the status flag CCF is set in the SAES_ISR register, then read the four data words from the SAES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
   d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).
3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the SAES peripheral by clearing EN.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the SAES_DINR register, to allow sending the key to the AES co-processor.

NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

The method uses interrupt from the SAES peripheral to control the data append, through the following sequence:

1. Enable interrupts from SAES, by setting the CCFIE bit of the SAES_IER register.
2. When KEYVALID is set, enable the SAES peripheral, by setting EN (if not already done).
3. Write first four input data words into the SAES_DINR register.
4. Handle the data in the SAES interrupt service routine. Upon each interrupt:
   a) Read four output data words from the SAES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the SAES_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the SAES peripheral by clearing EN and quit the interrupt service routine.
   e) Write next four input data words into the SAES_DINR register and quit the interrupt service routine.

Note: SAES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two SAES computations.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.
Appending data using DMA

With this method, all the transfers and processing are managed by DMA and SAES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.

2. Configure the DMA controller so as to transfer the data to process from the memory to the SAES peripheral input and the processed data from the SAES peripheral output to the memory, as described in Section 34.6: SAES DMA requests. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer must not include the last four-word block if padded with zeros. The sequence described in Appending data using the CPU in polling mode must be used instead for this last block, because the NPBLB[3:0] bitfield must be set up before processing the block, for SAES to compute a correct tag.

3. When KEYVALID is set, enable the SAES peripheral, by setting EN (if not already done).

4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.

5. Upon DMA interrupt indicating the transfer completion, get the SAES-processed data from the memory.

Note: The CCF flag has no use with this method because the reading of the SAES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.
34.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

**Figure 310. Typical operation with authentication**

---

Section 34.4.11: SAES Galois/counter mode (GCM) and Section 34.4.13: SAES counter with CBC-MAC (CCM) describe detailed sequences supported by SAES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by SAES is described in Section 34.4.12: SAES Galois message authentication code (GMAC).

34.4.7 SAES ciphertext stealing and data padding

When using SAES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques such as those described in NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode. Since SAES does not implement such techniques, the application must complete the last block of input data using data from the second last block.

*Note:* **Ciphertext stealing techniques are not documented in this reference manual.**

Similarly, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since SAES does not implement automatic data padding operation to the last block, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.
34.4.8 SAES suspend and resume operations

A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as SAES is enabled again, to receive a next data block. 

*Figure 311* gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

*Figure 311. Example of suspend mode management*

A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

34.4.9 SAES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB, a special key scheduling is required before processing the first block.

*Figure 312* and *Figure 313* describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.
In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC, a special key scheduling is required before processing the first block.

Figure 314 and Figure 315 describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.
**For more details, refer to NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.**

### ECB and CBC encryption process

This process is described in Section 34.4.5, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in encryption mode (write MODE[1:0] with 0x0).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to Section 34.4.14 (wrapped keys) and Section 34.4.15 (shared keys).
4. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to Section 34.4.17: SAES key registers for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in Section 34.4.7.
   b) Append the cleartext block into SAES as described in Section 34.4.5, then read the SAES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step b) until the third-last plaintext block is encrypted. For the last two blocks, follow the steps a) and b).
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.
ECB/CBC decryption process

This process is described in Section 34.4.5, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   - Select the key derivation mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to Section 34.4.14 (wrapped keys) and Section 34.4.15 (shared keys).
4. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to Section 34.4.17: SAES key registers for details.
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the SAES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.
7. Wait until the CCF flag in the SAES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register. The decryption key is available in the AES core and SAES is disabled automatically.
9. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in decryption mode (write MODE[1:0] with 0x2). Do not change other parameters.
10. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
11. Enable the SAES peripheral, by setting EN.
12. Append encrypted data:
   a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in Section 34.4.7.
   b) Append the ciphertext block into SAES as described in Section 34.4.5, then read the SAES_DOUTR register four times to save the cleartext block (MSB first).
   c) Repeat the step b) until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps a) and b).
13. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in ECB/CBC modes

The following sequences are valid for normal key mode (KMOD[1:0] at 0x0).

To suspend the processing of a message, proceed as follows:
1. If DMA is used, stop the SAES DMA transfers to the input FIFO, by clearing the DMAINEN bit of the SAES_CR register.
2. If DMA is not used, read four times the SAES_DOUTR register to save the last processed block. If DMA is used, wait until the CCF flag is set in the SAES_ISR
register then stop the DMA transfers from the output FIFO, by clearing the
DMAOUTEN bit of the SAES_CR register.
3. If DMA is not used, wait until the CCF flag in the SAES_ISR register is set (computation
completed).
4. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
5. Save initialization vector registers (only required in CBC mode as the SAES_IVRx
registers are altered during the data processing).
6. Disable the SAES peripheral, by clearing EN.
7. Save the SAES_CR register and clear the key registers if they are not needed, to
process the higher-priority message.
8. If DMA is used, save the DMA controller status (pointers for SAES input and output
data transfers, number of remaining bytes, and so on).

To resume the processing of a message, proceed as follows:
1. If DMA is used, configure the DMA controller so as to complete the remaining input
FIFO and output FIFO transfers.
2. Disable the SAES peripheral, by clearing EN.
3. Restore the SAES_CR register (with correct KEYSIZE) then restore the SAES_KEYRx
registers. Alternatively, select a key source different from key registers, using
KEYSEL[2:0]. Refer to Section 34.4.17: SAES key registers for details.
4. Prepare the decryption key, as described in ECB/CBC decryption process (only
required for ECB or CBC decryption).
5. Restore the SAES_IVRx registers, using the saved configuration (only required in CBC
mode).
6. Enable the SAES peripheral, by setting EN.
7. If DMA is used, enable SAES DMA transfers, by setting DMAINEN and DMAOUTEN.

Note: It is not required to save the key registers as the application knows the original key.

34.4.10 SAES counter (CTR) mode

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed
with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key
scheduling is required for the CTR decryption since the AES core is always used in
encryption mode.
A typical message construction in CTR mode is given in Figure 316.

**Figure 316. Message construction in CTR mode**

The structure of this message is:

- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.

- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

**CTR encryption and decryption**

*Figure 317* describes the counter (CTR) chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.

**Figure 317. CTR encryption**
Initialization vectors in SAES must be initialized as shown in Table 316.

Table 316. Counter mode initialization vector definition

<table>
<thead>
<tr>
<th>SAES_IVR3[31:0]</th>
<th>SAES_IVR2[31:0]</th>
<th>SAES_IVR1[31:0]</th>
<th>SAES_IVR0[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-bit counter = 0x0001</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CTR encryption and decryption process

This process is described in Section 34.4.5, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to Section 34.4.14 (wrapped keys) and Section 34.4.15 (shared keys).
4. Write the initialization vector into the SAES_IVRx registers according to Table 316.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to Section 34.4.17: SAES key registers for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into SAES as described in Section 34.4.5, then read the SAES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step b) until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps a) and b). For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in Section 34.4.9: SAES basic chaining modes (ECB, CBC).

Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.
34.4.11 SAES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

![Figure 318. Message construction in GCM](image)

The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV)**: a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.

- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} - 1$ bits. This part of the message is only authenticated, not encrypted.

- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed $2^{32} - 2$ 16-byte blocks.

- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in Table 318.

The GCM standard specifies that ciphertext C has the same bit length as the plaintext P. When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

For more details, refer to NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*
Figure 319 describes the GCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in Table 317.

<table>
<thead>
<tr>
<th>Table 317. Initialization of IV registers in GCM mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAES_IVR3[31:0]</td>
</tr>
<tr>
<td>32-bit counter = 0x0002</td>
</tr>
</tbody>
</table>

The last block of a GCM message contains the AAD header length and the payload length information, as shown in Table 318.

<table>
<thead>
<tr>
<th>Table 318. GCM last block definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word order to SAES_DINR</td>
</tr>
</tbody>
</table>
GCM encryption and decryption process

This process is described in Section 34.4.6, with the following sequence of events:

**GCM initialize**

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (such as secure, non-secure, specific CPU), also set KEYPROT.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to Section 34.4.14 (wrapped keys) and Section 34.4.15 (shared keys).
   - Select the GCM initialization phase, by writing GCMPH[1:0] with 0x0.
4. Write the initialization vector in SAES_IVRx registers according to Table 317.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to Section 34.4.17: SAES key registers for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
8. Wait until the CCF flag is set in the SAES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
9. Clear the CCF flag by setting the CCF bit of the SAES_ICR register.

**GCM header phase**

10. Initialize header phase:
   a) Select the GCM header phase, by writing 0x1 to GCMPH[1:0]. Do not change the other configurations written during GCM initialization.
   b) Enable the SAES peripheral, by setting EN.
11. Append header data:
   a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into SAES as described in Section 34.4.5.
   c) Repeat the step b) until the second-last AAD data block is processed. For the last block, follow the steps a) and b).

**Note:** This phase can be skipped if there is no AAD, that is, Len(A) = 0.
No data are read during header phase.
GCM payload phase
12. Initialize payload phase:
   a) Select the GCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
   b) If the header phase is skipped, enable the SAES peripheral by setting EN.
13. Append payload data:
   a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into SAES as described in Section 34.4.5, then read the SAES_DOUTR register four times to save the resulting block
   c) Repeat the step b) until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps a) and b). For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).

GCM finalization
14. Encryption only: wait until the BUSY flag in the SAES_SR register is cleared.
15. Select the GCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during GCM initialization.
16. Write the final GCM block into the SAES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in Table 318.
17. Wait until the CCF flag in the SAES_ISR register is set.
18. Get the GCM authentication tag, by reading the SAES_DOUTR register four times.
19. Clear the CCF flag, by setting the CCF bit in SAES_ICR register.
20. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to SAES_DINR normally (no swapping), while swapping is applied to tag data read from SAES_DOUTR.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode
Suspend/resume operations are not supported in GCM mode.

34.4.12 SAES Galois message authentication code (GMAC)
The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.
For more details, refer to NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC. Figure 321 describes the GMAC chaining implementation in the SAES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

**Figure 321. GMAC authentication mode**

The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

**Suspend/resume operations in GMAC**

Suspend/resume operations are not supported in GMAC mode.
34.4.13 SAES counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

**Figure 322. Message construction in CCM mode**

The structure of the message is:

- **16-byte first authentication block (B0),** composed of three distinct fields:
  - Q: a bit string representation of the octet length of P (Len(P))
  - Nonce (N): a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - Flags: most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values t (MAC length expressed in bytes) and Q (plaintext length such that Len(P) < 2^Q bytes). The counter blocks range associated to Q is equal to 2^Q-4, that is, if the maximum value of Q is 8, the counter blocks used in cipher must be on 60 bits.

- **16-byte blocks (B) associated to the associated data (A).**

  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see Figure 322). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:
  - If 0 < a < 2^16 - 2^5, then it is encoded as [a]16, that is, on two bytes.
  - If 2^16 - 2^8 < a < 2^32, then it is encoded as 0xff || 0xfe || [a]32, that is, on six bytes.
  - If 2^32 < a < 2^64, then it is encoded as 0xff || 0xff || [a]64, that is, on ten bytes.
- **16-byte blocks (B)** associated to the plaintext message \( P \), which is both authenticated and encrypted as ciphertext \( C \), with a known length \( \text{Len}(P) \). This length can be a non-multiple of 16 bytes (see Figure 322).

- **Encrypted MAC (T)** of length \( \text{Len}(T) \) appended to the ciphertext \( C \) of overall length \( \text{Len}(C) \).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

*Note:* CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

- \( N: 10111213\ 141516 \) (Len(N) = 56 bits or 7 bytes)
- \( A: 00010203\ 04050607 \) (Len(A) = 64 bits or 8 bytes)
- \( P: 20212223 \) (Len(P) = 32 bits or 4 bytes)
- \( T: 6084341B \) (Len(T) = 32 bits or \( t = 4 \))
- \( B0: 4F101112\ 13141516\ 00000000\ 00000004 \)
- \( B1: 00080001\ 02030405\ 06070000\ 00000000 \)
- \( B2: 20212223\ 00000000\ 00000000\ 00000000 \)
- \( \text{CTR}0: 0710111213\ 141516\ 00000000\ 00000000 \)
- \( \text{CTR}1: 0710111213\ 141516\ 00000000\ 00000001 \)

For more details, refer to NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality.
Figure 323 describes the CCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

The first block of a CCM message (B0) must be prepared by the application as defined in Table 319.

Table 319. Initialization of IV registers in CCM mode

<table>
<thead>
<tr>
<th>SAES_IVR3[31:0]</th>
<th>SAES_IVR2[31:0]</th>
<th>SAES_IVR1[31:0]</th>
<th>SAES_IVR0[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0[127:96](^{(1)})</td>
<td>B0[95:64]</td>
<td>B0[63:32]</td>
<td>B0[31:0](^{(2)})</td>
</tr>
</tbody>
</table>

1. The 5 most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

SAES supports counters up to 64 bits, as specified by NIST.
CCM encryption and decryption process

This process is described in Section 34.4.6, with the following sequence of events:

**CCM initialize**

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context, also set the KEYPROT bit.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to Section 34.4.14 (wrapped keys) and Section 34.4.15 (shared keys).
   - Select the CCM initialization phase, by writing GCMPH[1:0] with 0x0.
4. Write the B0 data in SAES_IVRx registers according to Table 319.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to Section 34.4.17: SAES key registers for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
8. Wait until the CCF flag in the SAES_ISR register is set.
9. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.

**CCM header phase**

10. Initialize header phase:
    a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
    b) Select the CCM header phase, by writing GCMPH[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
    c) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in Section 34.4.5.
    c) Repeat the step b) until the second-last associated data block is processed. For the last block, follow the steps a) and b).

*Note:* This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.
CCM payload phase

12. Initialize payload phase:
   a) Select the CCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
   b) If the header phase is skipped, enable the SAES peripheral, by setting EN.

13. Append payload data:
   a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into SAES as described in Section 34.4.5, then read the SAES_DOUTR register four times to save the resulting block.
   c) Repeat the step b) until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps a) and b). For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).
Remove LSB_{Len(T)}(C) encrypted tag information when decrypting ciphertext C.

CCM finalization

14. Select the CCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.

15. Wait until CCF flag in the SAES_ISR register is set.

16. Get the CCM authentication tag, by reading the SAES_DOUTR register four times.

17. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.

18. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from SAES_DOUTR register.
When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

To suspend the processing of a message in header or payload phase, proceed as follows:

1. If DMA is used, stop the SAES DMA transfers to the input FIFO, by clearing DMAINEN. If DMA is not used, wait until the CCF flag of the SAES_ISR register is set (computation completed).

2. In the payload phase, if DMA is not used, read four times the SAES_DOUTR register to save the last-processed block. If DMA is used, wait until the CCF flag in the SAES_ISR register is set, then stop the DMA transfers from the output FIFO, by clearing DMAOUTEN.

3. Clear the CCF flag in the SAES_ISR register, by setting the CCF bit of the SAES_ICR register.

4. Save the SAES_SUSPRx registers in the memory.

5. Save the IV registers as they are altered during the data processing.

6. Disable the SAES peripheral, by clearing EN.
7. Save the current SAES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

8. If DMA is used, save the DMA controller status (pointer for SAES input data transfers, number of remaining bytes, and so on). In the payload phase, also save pointer for SAES output data transfers.

To resume the processing of a message, proceed as follows:
1. If DMA is used, configure the DMA controller in order to complete the remaining input FIFO transfers. In the payload phase, also configure the DMA controller for the remaining output FIFO transfers.
2. Disable the SAES peripheral, by clearing EN.
3. Write the suspend register values, previously saved in the memory, back into their corresponding SAES_SUSPRx registers.
4. Restore SAES_IVRx registers using the saved configuration.
5. Restore the initial setting values in the SAES_CR and SAES_KEYRx registers. Alternatively, select a key source different from the key registers, by setting KEYSEL[2:0] to a value different from 0x0. Refer to Section 34.4.17: SAES key registers for details.
6. Enable the SAES peripheral, by setting EN.
7. If DMA is used, enable SAES DMA requests, by setting DMAINEN (and DMAOUTEN if in payload phase).

34.4.14 SAES operation with wrapped keys

SAES peripheral can wrap (encrypt) and unwrap (decrypt) application keys using hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, AES keys can be made usable by application software without being exposed in clear-text (unencrypted).

Wrapped key sequences are too small to be suspended/resumed. SAES cannot unwrap a key using an unwrapped key.

Operation with wrapped keys for SAES in ECB and CBC modes

*Figure 324* summarizes how to wrap or unwrap keys for SAES in ECB and CBC modes. To protect the wrapped key, select DHUK by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.
The recommended sequence to wrap (that is, encrypt) a key is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in encryption mode (MODE[1:0] at 0x0)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. This information is used both for the encryption key and for the key to be encrypt.
   - Select wrapped key mode by writing KMOD[1:0] with 0x1
4. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1 or 0x4. Refer to Section 34.4.17 for details on the use of KEYSEL[2:0] at 0x4.
6. Wait until KEYVALID is set (DHUK loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Write the SAES_DINRx register four times to input the key to encrypt (MSB first, see Table 321 on page 1329).
9. Wait until CCF flag is set in the SAES_ISR register.
10. Get the encrypted key (MSB first) by reading the SAES_DOUTRx register four times. Then clear the CCF flag, by setting the CCF bit in SAES_ICR register.
11. Repeat steps 8. to 10. if KEYSIZE is set.
12. Disable the SAES peripheral, by clearing EN.

Note: Encryption in Wrapped-key mode is only supported when ECB or CBC is selected.
Key unwrapping for SAES

The recommended sequence to unwrap (or decrypt) a wrapped (encrypted) key using ECB/CBC is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the chaining mode used during the wrapping process (CHMOD[2:0] at 0x0 or 0x1) in key derivation mode (MODE[1:0] at 0x1)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size used during the wrapping process, with KEYSIZE. This information is used both for the decryption key and for the key to decrypt.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. Select the DHUK key source, by writing KEYSEL[2:0] with 0x1 or 0x4. Refer to Section 34.4.17 for details on the use of KEYSEL[2:0] at 0x4.
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN bit in SAES_CR to enable the peripheral
7. Wait until CCF flag is set in the SAES_ISR register.
8. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The decryption key is available in the AES core, and SAES is disabled automatically.
9. Select the decryption mode (MODE[1:0] at 0x2). Other parameters are unchanged.
10. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
11. Enable the SAES peripheral, by setting EN.
12. Write the SAES_DINR register four times to input the key to decrypt (MSB first, see Table 321 on page 1329).
13. Wait until CCF flag is set in the SAES_ISR register. Then clear the CCF flag by setting the CCF bit in SAES_ICR register. Reading SAES_DOUTR returns zero and triggers a read error (RDERR).
15. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the decrypted wrapped key is immediately usable by the application for any AES operation (normal key mode). Decrypted wrapped key can be shared with an application running in a different security context (different security or Compartment ID attribute) if KEYPROT bit was cleared during step 3.

Note: When KMOD[1:0] = 0x1 (wrapped key) and MODE[1:0] = 0x2 (decryption) a read access to SAES_DOUTR register triggers a read error (RDERR).

When KEYSEL[2:0] is at 0x1 (DHUK) or 0x4 (DHUK XOR BHK), the application software must use the same privilege, security, KMOD[1:0], CHMOD[2:0] and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

Operation with wrapped keys for SAES in CTR mode

Figure 325 summarizes how to unwrap keys for SAES in CTR mode. To protect the derived key, select DHUK by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.
The recommended sequence for SAES wrapped key mode using CTR is as follows:
1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the CTR chaining mode (CHMOD[2:0] at 0x2) in decryption mode (MODE[1:0] at 0x2). Other MODE[1:0] values are not supported.
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. It is used for encryption key and for the key to share.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. Write the initialization vector in SAES_IVRx registers, keeping the two least significant bits of SAES_IVR0 at zero.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1 or 0x4. Refer to Section 34.4.17 for details on the use of KEYSEL[2:0] at 0x4.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Wait until CCF flag is set in the SAES_ISR register.
9. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The derived hardware secret key is available in SAES_KEYRx registers.
10. Repeat steps 8. and 9. if KEYSIZE is set.
11. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the hardware secret key derived from the public data in the SAES_IVRx registers is then immediately usable by the application for any AES operation (normal key mode).
**Note:** The configuration KMOD[1:0] at 0x1 (wrapped key), CHMOD[2:0] at 0x2 (CTR chaining), and MODE at 0x0 (encryption) disables the peripheral, by automatically clearing the EN bit of the SAES_CR register.

### 34.4.15 SAES operation with shared keys

SAES peripheral can share application keys wrapped with hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, the application software can make the AES keys available to the AES peripheral, without exposing them in clear-text (unencrypted).

Shared key sequences are too small to be suspended/resumed. SAES cannot unwrap a shared key using an unwrapped key.

**Note:** When a key stored in SAES is shared with AES, the protection given by KEYPROT bit is lost. The protection is detailed in Section 34.4.17: SAES key registers.

*Figure 326* summarizes how to wrap or unwrap keys to share with AES peripheral. To protect the shared key, DHUK must be selected, by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

**Figure 326. Usage of Shared-key mode**

<table>
<thead>
<tr>
<th>Step 1: provision</th>
<th>Step 2: load</th>
<th>Step 3: use</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIN</td>
<td>DIN</td>
<td>DIN</td>
</tr>
<tr>
<td>clear-text shared key</td>
<td>wrapped shared key</td>
<td>wrapped (encrypted) shared key</td>
</tr>
<tr>
<td>DHUK</td>
<td>DHUK</td>
<td>DHUK</td>
</tr>
<tr>
<td>BHK + DHUK</td>
<td>BHK + DHUK</td>
<td>BHK + DHUK</td>
</tr>
<tr>
<td>DOUT</td>
<td>DOUT</td>
<td>DOUT</td>
</tr>
<tr>
<td>wrapped (encrypted) shared key</td>
<td>IV (if applicable)</td>
<td>shared key</td>
</tr>
<tr>
<td>enc</td>
<td>enc/dec</td>
<td>enc/dec</td>
</tr>
<tr>
<td><strong>SAES</strong> enc</td>
<td><strong>SAES</strong> enc/dec</td>
<td><strong>SAES</strong> enc/dec</td>
</tr>
<tr>
<td><strong>KEYSEL</strong> = 001 or 100</td>
<td><strong>KEYSEL</strong> = 001 or 100</td>
<td><strong>KEYSEL</strong> = 00 (to keep the wrapped key)</td>
</tr>
<tr>
<td><strong>MODE</strong> = 00 (encryption)</td>
<td><strong>MODE</strong> = 10 (decryption)</td>
<td><strong>MODE</strong> = any value</td>
</tr>
</tbody>
</table>

**Note:** DHUK value depends on privilege, KMOD[1:0], KSHAREID, KEYSEL[2:0], CHMOD[2:0], and on whether SAES peripheral is secure or non-secure.

In the step 3, AES represents the AES peripheral.

### Key wrapping for AES peripheral

Before SAES can share a key with the AES peripheral, the key must be encrypted (wrapped) once. The encryption sequence of a shared key is the same as for a wrapped key, with KMOD[1:0] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in *Figure 326*. See Key wrapping for SAES for details.

**Note:** Encryption in Shared-key mode is only supported when ECB or CBC is selected.
Key unwrapping for AES peripheral (shared key)

Each time SAES needs to share a key with the AES peripheral, shared encrypted key must be decrypted (unwrapped) in SAES, then loaded by AES. The overall sequence is described next.

Sequence in the SAES peripheral

The decryption sequence of a shared key is the same as for a wrapped key, with KMOD[1:0] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in Figure 326. See Key unwrapping for SAES for details.

In shared key mode when decryption mode is selected (MODE[1:0] at 0x2), a read access to the SAES_DOUTR register triggers a read error (RDERR).

Note: Instead of being shared, a decrypted shared key can be used directly in SAES as the KEYSEL[2:0] bitfield is automatically cleared. In this case, KMOD[1:0] must be written with 0x0 (normal key mode).

Sequence in the AES peripheral

Once the shared key is decrypted in SAES key registers, it can be shared with the AES peripheral, while SAES peripheral remains in key sharing state, that is, with KMOD[1:0] at 0x2 and KEYVALID set. The sequence in the AES key share target peripheral is described in AES key sharing with secure AES co-processor of the corresponding section in this document. It can be run multiple times (for example, to manage a suspend/resume situation) as long as SAES is unused and duly remains in key sharing state.

Note: When KMOD[1:0] is at 0x2 and BUSY set in the AES peripheral, and KEYSIZE value of AES and SAES differs, the key sharing fails and the KEIF flag is raised in both peripherals.

When KEYSEL[2:0] is at 0x1 (DHUK) or 0x4 (DHUK XOR BHK), the application software must use the same privilege, security, KMOD[1:0] / KSHAREID[1:0], CHMOD[2:0], and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

34.4.16 SAES data registers and data swapping

Data input and output

A 16-byte data block enters the SAES peripheral with four successive 32-bit word writes into the SAES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the SAES peripheral with four successive 32-bit word reads of the SAES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See Table 320 “no swapping” option for details.

Data swapping

The SAES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the SAES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the SAES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.
The data swap type is selected through DATATYPE[1:0]. The selection applies to both SAES input and output.

Note: The data in SAES key registers (SAES_KEYRx) and initialization vector registers (SAES_IVRx) are not sensitive to the swap mode selection.

The SAES data swapping feature is summarized in Table 320 and Figure 327.

Table 320. AES data swapping example

<table>
<thead>
<tr>
<th>DATATYPE[1:0]</th>
<th>Swapping performed</th>
<th>Data block</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>System memory data (big-endian)</td>
</tr>
<tr>
<td>0x0</td>
<td>No swapping</td>
<td>Block[127..64]: 0x04EEF672 2E04CE96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Block[63..0]: 0x4E6F7720 69732074</td>
</tr>
<tr>
<td>0x1</td>
<td>Half-word (16-bit) swapping</td>
<td>Block[63..0]: 0x4E6F 7720 6973 2074</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @, word[63..32]: 0x7720 4E6F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0x4, word[31..0]: 0x2074 6973</td>
</tr>
<tr>
<td>0x2</td>
<td>Byte (8-bit) swapping</td>
<td>Block[63..0]: 0x4E 6F 77 20 69 73 20 74</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @, word[63..32]: 0x2077 6F4E</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0x4, word[31..0]: 0x7420 7369</td>
</tr>
<tr>
<td>0x3</td>
<td>Bit swapping</td>
<td>Block[63..32]: 0x4E6F7720</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0100 1110 0110 1111 0111 0111 0010 0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Block[31..0]: 0x69732074</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0110 1001 0111 0011 0010 0000 0111 0100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @, word[63..32]: 0x04EE F672</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0000 0100 1110 1110 1111 0110 0111 0010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0x4, word[31..0]: 0x2E04 CE96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0010 1110 0000 0100 1100 1110 1001 0110</td>
</tr>
</tbody>
</table>
Data padding

Figure 327 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3
34.4.17 SAES key registers

The eight SAES_KEYR[0] write-only registers store the encryption or decryption key information, as shown on Table 321. Reads are not allowed for security reason.

**Note:** In memory and in SAES key registers, keys are stored in little-endian format, with most significant byte on the highest address.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>KEY[127:96]</td>
<td>KEY[95:64]</td>
<td>KEY[63:32]</td>
<td>KEY[31:0]</td>
</tr>
</tbody>
</table>

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the SAES_KEYR[0] registers are ignored when SAES peripheral is enabled (EN bit set) and KEYSEL[2:0] is different from zero. The application must check this before modifying key registers.

The entire key must be written before starting an AES computation. In normal key mode (KMOD[1:0] at 0x0), with KEYSEL[2:0] at 0x0, the key registers must always be written in either ascending or descending order. The write sequence becomes:

- SAES_KEYR[0] (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- SAES_KEYR[0] (x = 0 to 7 or x=7 to 0) for KEYSIZE set

**Note:** KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see Section 34.4.19).

**Key selection**

With KEYSEL[2:0] at 0x0, the application must write the key in the SAES_KEYR[0] registers.

With KEYSEL[2:0] at 0x1, a derived hardware unique key (DHUK), computed inside SAES from a non-volatile and secret root hardware unique key, is loaded directly into key registers, based on KEYSIZE information. Thanks to the key derivation function, a secure SAES uses a secure DHUK, while a non-secure SAES uses a non-secure DHUK.

With KEYSEL[2:0] at 0x2, the boot hardware key (BHK), stored in tamper-resistant secure backup registers, is entirely transferred into key registers upon a secure application performing a single read of all TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE cleared, x = 0 to 7 for KEYSIZE set) in either ascending or descending order. Refer to Table 321.

With KEYSEL[2:0] at 0x4, the XOR combination of DHUK and BHK is entirely transferred into key registers upon a secure application performing a single read of all TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE cleared, x = 0 to 7 for KEYSIZE set) in either ascending or descending order. Refer to Table 321.
Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK if KEYVALID is set. The recommended method to clear KEYVALID is to set IPRST. Such method is required for example when switching from ECB decryption to ECB encryption, selecting the same BHK (KEYSEL[2:0] at 0x2).

For all KEYSEL[2:0] values, initiating the key-loading sequence sets the BUSY flag and clears the KEYVALID flag. Once the amount of bits defined by KEYSIZE is transferred to the SAES_KEYRx registers, BUSY is cleared, KEYVALID set and the EN bit becomes writable. If an error occurs, BUSY and KEYVALID are cleared and KEIF set (see Section 34.4.19).

**Key protection**

As depicted in Figure 328, when an application sets the KEYPROT bit before writing a key in SAES_KEYRx, any application executing in a different security context (that is, different security attribute) triggers a KEIF error flag upon access to any SAES register when KEYVALID is set.

**Note:** KEYSEL[2:0] values different from zero (normal key) automatically protect the key registers.

**Figure 328. Key protection mechanisms**

![Key protection mechanisms](image)

34.4.18 SAES initialization vector registers

The four SAES_IVRx registers store the initialization vector (IV) information, as shown in Table 322. They can only be written if the SAES peripheral is disabled (EN cleared).

**Note:** In memory and in SAES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

<table>
<thead>
<tr>
<th>SAES_IVR3[31:0]</th>
<th>SAES_IVR2[31:0]</th>
<th>SAES_IVR1[31:0]</th>
<th>SAES_IVR0[31:0]</th>
</tr>
</thead>
</table>
Initialization vector information depends on the chaining mode selected. When used, SAES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

### 34.4.19 SAES error management

The SAES peripheral manages the errors described in this section.

**Read error flag (RDERRF)**

Unexpected read attempt of the SAES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

*Note:* Unless otherwise indicated, **SAES is not disabled when RDERRF rises and it continues processing.**

An interrupt is generated if the RWEIE bit is set. For more details, refer to [Section 34.5: SAES interrupts](#).

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

**Write error flag (WDERR)**

Unexpected write attempt of the SAES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

*Note:* Unless otherwise indicated, **SAES is not disabled when WRERRF rises and it continues processing.**

An interrupt is generated if the RWEIE bit is set. For more details, refer to [Section 34.5: SAES interrupts](#).

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

**Key error interrupt flag (KEIF)**

There are multiple sources of errors that set the KEIF flag of the SAES_ISR register and clear the KEYVALID bit of the SAES_SR register:

- **Key protection error:** while KEYVALID is set, then if KEYPROT is set or KEYSEL[2:0] is different from zero, this error is triggered when an application executing in a security context different from the one used to load the key (that is, different security attribute) accesses SAES.

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See [Section 34.4.17: SAES key registers](#) for details.

- **Key sharing size mismatch error:** triggered when KMOD[1:0] is at 0x2 and KEYSIZE in AES peripheral does not match KEYSIZE in SAES peripheral.
Secure AES coprocessor (SAES)

- **Key sharing error**: triggered upon failing transfer of SAES shared key to AES peripheral. See Section 34.4.15: SAES operation with shared keys for details.

- **Hardware secret key loading error**: triggered upon failing load of DHUK or BHK into SAES. KEYSEL[2:0] at 0x1 (DHUK), 0x2 (BHK) or 0x4 (DHUK XOR BHK) is not functional.

The KEIF flag is cleared with corresponding bit of the SAES_ICR register. An interrupt is generated if the KEIE bit of the SAES_IER register is set. For more details, refer to Section 34.5: SAES interrupts.

Upon a key selection error, clearing the KEIF flag automatically restarts the key selection process. Persisting problems (for example, RHUK load failing) may require a power-on reset.

Upon a key sharing error, reset both AES and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

**Note:** For any key error, clear KEIF flag prior to disabling and re-configuring SAES.

### RNG error interrupt flag (RNGEIF)

SAES fetches random numbers from the RNG peripheral automatically after an IP reset triggered in the RCC. SAES cannot be used when RNGEIF is set.

An error detected while fetching a random number from RNG peripheral (due to, for example, bad entropy) sets the RNGEIF flag of the SAES_ISR register. The flag is cleared by setting the corresponding bit of the SAES_ICR register. An interrupt is generated if the RNGEIF bit of the SAES_IER register is set. For more details, refer to Section 34.5: SAES interrupts.

Upon an RNG error:

- Verify that the RNG peripheral AHB clock is enabled and no noise source (or seed) error is pending in this peripheral.
- Clear RNGEIF or reset the peripheral by setting IPRST. The clearance of the BUSY flag then indicates the completion of the random number fetch from RNG.

**Note:** To avoid RNGEIF errors, it is recommended to activate the RNG AHB clock each time SAES AHB clock is activated.

### About DPA errors

An unexpected error triggers an SAES internal tamper event in the TAMP peripheral, and stops any SAES co-processor processing.

To resume normal operation, reset the SAES peripheral through RCC or global reset.
34.5 **SAES interrupts**

There are multiple individual maskable interrupt sources generated by the SAES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)
- RNG error (RNGEIF)

See Section 34.4.19: SAES error management for details on SAES errors.

These sources are combined into a common interrupt signal from the SAES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable SAES interrupt sources individually by setting/clearing the corresponding enable bit of the SAES_IER register.

The status of the individual maskable interrupt sources can be read from the SAES_ISR register. They are cleared by setting the corresponding bit of the SAES_ICR register.

Table 323 gives a summary of the available features.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAES</td>
<td>computation completed flag</td>
<td>CCF</td>
<td>CCFIE</td>
<td>set CCF(^{(1)})</td>
</tr>
<tr>
<td></td>
<td>read error flag</td>
<td>RDERRF(^{(2)})</td>
<td>RWEIE</td>
<td>set RWEIE(^{(1)})</td>
</tr>
<tr>
<td></td>
<td>write error flag</td>
<td>WRERRF(^{(2)})</td>
<td>KEIE</td>
<td>set KEIF(^{(1)})</td>
</tr>
<tr>
<td></td>
<td>key error flag</td>
<td>KEIF</td>
<td>KEIE</td>
<td>set KEIF(^{(1)})</td>
</tr>
<tr>
<td></td>
<td>RNG error flag</td>
<td>RNGEIF</td>
<td>RNGEIE</td>
<td>set RNGEIE(^{(1)})</td>
</tr>
</tbody>
</table>

1. Bit of the SAES_ICR register.
2. Flag of the SAES_SR register, mirrored by the flag RWEIF of the SAES_ISR register.

34.6 **SAES DMA requests**

The SAES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the SAES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

SAES only supports single DMA requests.

Detailed usage of DMA with SAES can be found in Appending data using DMA subsection of Section 34.4.5: SAES encryption or decryption typical usage.

**Data input using DMA**

Setting DMAINEN enables DMA writing into SAES. SAES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the SAES_DINR register (quadruple 32-bit word, MSB first).
Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 34.4.7).

Data output using DMA

Setting DMAOUTEN enables DMA reading from SAES. SAES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the SAES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, SAES switches automatically to a new input phase for the next data block, if any.

In DMA mode, the CCF flag has no use because the reading of the SAES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to managing the data transfers by software.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when SAES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

34.7 SAES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

<table>
<thead>
<tr>
<th>Key size</th>
<th>Mode of operation</th>
<th>Chaining algorithm</th>
<th>Clock cycles(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128-bit</td>
<td>Encryption or decryption(2)</td>
<td>ECB, CBC, CTR</td>
<td>480</td>
</tr>
<tr>
<td></td>
<td>Key preparation</td>
<td>-</td>
<td>145</td>
</tr>
<tr>
<td>256-bit</td>
<td>Encryption or decryption(2)</td>
<td>ECB, CBC, CTR</td>
<td>680</td>
</tr>
<tr>
<td></td>
<td>Key preparation</td>
<td>-</td>
<td>230</td>
</tr>
</tbody>
</table>

1. SAES kernel clock
2. Excluding key preparation time (ECB and CBC only).

<table>
<thead>
<tr>
<th>Key size</th>
<th>Mode of operation</th>
<th>Chaining algorithm</th>
<th>Initialization phase</th>
<th>Header phase(1)</th>
<th>Payload phase(1)</th>
<th>Final phase(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128-bit</td>
<td>Mode 1: Encryption/Mode 3: Decryption</td>
<td>GCM</td>
<td>490</td>
<td>72(2)</td>
<td>480(3)</td>
<td>490</td>
</tr>
<tr>
<td></td>
<td>Mode 3: Decryption</td>
<td>CCM</td>
<td>490</td>
<td>490</td>
<td>800</td>
<td>490</td>
</tr>
<tr>
<td>256-bit</td>
<td>Mode 1: Encryption/Mode 3: Decryption</td>
<td>GCM</td>
<td>650</td>
<td>72(2)</td>
<td>690(3)</td>
<td>650</td>
</tr>
<tr>
<td></td>
<td>Mode 3: Decryption</td>
<td>CCM</td>
<td>650</td>
<td>680</td>
<td>1350</td>
<td>650</td>
</tr>
</tbody>
</table>

1. Data insertion can include wait states forced by SAES on the AHB bus (maximum 3 cycles, typical 1 cycle).
2. SAES AHB clock cycles instead of kernel clock cycle (Galois multiplier only).
3. As a worst case in encryption mode, add extra 72 AHB clock cycles for the last block computation.
34.8 SAES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

34.8.1 SAES control register (SAES_CR)

Address offset: 0x000

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>IPRST: SAES peripheral software reset</td>
</tr>
<tr>
<td>Setting the bit resets the SAES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data are lost. For this reason, it is recommended to set the bit before handing over the SAES to a less secure application. The bit must be kept low while writing any configuration registers.</td>
<td></td>
</tr>
<tr>
<td>30:28</td>
<td>KEYSEL[2:0]: Key selection</td>
</tr>
<tr>
<td>The bitfield defines the source of the key information to use in the AES cryptographic core.</td>
<td></td>
</tr>
<tr>
<td>0x0: Software key, loaded in key registers SAES_KEYx</td>
<td></td>
</tr>
<tr>
<td>0x1: Derived hardware unique key (DHUK)</td>
<td></td>
</tr>
<tr>
<td>0x2: Boot hardware key (BHK)</td>
<td></td>
</tr>
<tr>
<td>0x4: XOR of DHUK and BHK</td>
<td></td>
</tr>
<tr>
<td>Others: Reserved (if used, unfreeze SAES with IPRST)</td>
<td></td>
</tr>
<tr>
<td>When KEYSEL[2:0] is different from zero, selected key value is available in key registers when BUSY bit is cleared and KEYVALID is set in the SAES_SR register. Otherwise, the key error flag KEIF is set. Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK when KEYVALID is cleared.</td>
<td></td>
</tr>
<tr>
<td>When the application software changes the key selection by writing the KEYSEL[2:0] bitfield, the key registers are immediately erased and the KEYVALID flag cleared.</td>
<td></td>
</tr>
<tr>
<td>At the end of the decryption process, if KMOD[1:0] is other than zero, KEYSEL[2:0] is cleared.</td>
<td></td>
</tr>
<tr>
<td>With the bitfield value other than zero and KEYVALID set, the application cannot transfer the ownership of SAES with a loaded key to an application running in another security context (such as secure, non-secure). More specifically, when security of an access to any register does not match the information recorded by SAES, the KEIF flag is set. Attempts to write the bitfield are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_CR register is set before the write access and it is not cleared by that write access.</td>
<td></td>
</tr>
</tbody>
</table>
Bits 27:26 **KSHAREID[1:0]**: Key share identification

This bitfield defines, at the end of a decryption process with KMOD[1:0] at 0x2 (shared key), which target can read the SAES key registers using a dedicated hardware bus.

0x0: AES peripheral

Others: Reserved

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 25:24 **KMOD[1:0]**: Key mode selection

The bitfield defines how the SAES key can be used by the application. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.

0x0: Normal key mode. Key registers are freely usable and no specific use or protection applies to SAES_DINR and SAES_DOUTR registers.

0x1: Wrapped key for SAES mode. Key loaded in key registers can only be used to encrypt or decrypt AES keys. Hence, when a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.

0x2: Shared key mode. After a successful decryption process (unwrapping), SAES key registers are shared with the peripheral described in KSHAREID[1:0] bitfield. This sharing is valid only while KMOD[1:0] at 0x2 and KEYVALID=1. When a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.

Others: Reserved

With KMOD[1:0] other than zero, any attempt to configure the SAES peripheral for use by an application belonging to a different security domain (such as secure or non-secure) results in automatic key erasure and setting of the KEIF flag.

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 23:20 **NPBLB[3:0]**: Number of padding bytes in last block

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)

0x1: Padding for the last LSB byte

... 0xF: Padding for the 15 LSB bytes of last block.

Bit 19 **KEYPROT**: Key protection

When set, hardware-based key protection is enabled.

0: When KEYVALID is set and KEYSEL[2:0]=0 application can transfer the ownership of the SAES, with its loaded key, to an application running in another security context (such as non-secure, secure).

1: When KEYVALID is set, key error flag (KEIF) is set when an access to any registers is detected, this access having a security context (for example, secure, non-secure) that does not match the one of the application that loaded the key.

Attempts to write the bit are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 18 **KEYSIZE**: Key size selection

This bitfield defines the key length in bits of the key used by SAES.

0: 128-bit

1: 256-bit

When KMOD[1:0] is at 0x1 or 0x2, KEYSIZE also defines the length of the key to encrypt or decrypt. Attempts to write the bit are ignored when BUSY is set, as well as when the EN is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.
Bit 15  Reserved, must be kept at reset value.

Bits 14:13  **GCMPH[1:0]**  GCM or CCM phase selection
   This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.
   0x0: Initialization phase
   0x1: Header phase
   0x2: Payload phase
   0x3: Final phase

Bit 12  **DMAOUTEN**: DMA output enable
   This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from SAES via DMA.
   0: Disable
   1: Enable
   Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11  **DMAINEN**: DMA input enable
   This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to SAES via DMA.
   0: Disable
   1: Enable
   Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7  Reserved, must be kept at reset value.

Bits 16, 6:5  **CHMOD[2:0]**  Chaining mode
   This bitfield selects the AES chaining mode:
   0x0: Electronic codebook (ECB)
   0x1: Cipher-block chaining (CBC)
   0x2: Counter mode (CTR)
   0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)
   others: Reserved
   Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 4:3  **MODE[1:0]**  Operating mode
   This bitfield selects the SAES operating mode:
   0x0: Encryption
   0x1: Key derivation (or key preparation), for ECB/CBC decryption only
   0x2: Decryption
   0x3: Reserved
   Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 2:1  **DATATYPE[1:0]**  Data type
   This bitfield defines the format of data written in the SAES_DINR register or read from the SAES_DOUTR register, through selecting the mode of data swapping. This swapping is defined in Section 34.4.16: SAES data registers and data swapping.
   0x0: No swapping (32-bit data).
   0x1: Half-word swapping (16-bit data)
   0x2: Byte swapping (8-bit data)
   0x3: Bit-level swapping
   Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.
Bit 0  **EN**: Enable
This bit enables/disables the SAES peripheral.
0: Disable
1: Enable
At any moment, clearing then setting the bit re-initializes the SAES peripheral. When KMOD[1:0] is different from 0x0, using IPRST bit is recommended instead.
This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.
The bit cannot be set as long as KEYVALID is cleared, or when SAES is in one of the following configurations:
– KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x3 (GCM)
– KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x2 (CTR), MODE[1:0] at 0x0 (encryption).

34.8.2 **SAES status register (SAES_SR)**
Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**RES.** Reserved, must be kept at reset value.

**Bit 7** **KEYVALID**: Key valid flag
This bit is set by hardware when the key of size defined by KEYSIZE is loaded in SAES_KEYRx key registers.
0: Key not valid
1: Key valid
The EN bit can only be set when KEYVALID is set.
In normal mode when KEYSEL[2:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.
When KEYSEL[2:0] is different from zero, the BUSY flag is automatically set by SAES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is set, BUSY cleared and KEYVALID remains cleared.
If set, KEIF must be cleared through the SAES_ICR register, otherwise KEYVALID cannot be set.
See the KEIF flag description for more details.
For further information on key loading, refer to Section 34.4.17: SAES key registers.

**Bits 6:4** Reserved, must be kept at reset value.
### Bit 3 BUSY: Busy
This flag indicates whether SAES is idle or busy.
0: Idle
1: Busy
SAES is flagged as idle when disabled (when EN is low) or when the last processing is completed. SAES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), fetching random number from the RNG, or transferring a shared key to the target peripheral. When GCM encryption is selected, this flag must be at zero before suspending current process to manage a higher-priority message. BUSY must also be cleared before selecting the GCM final phase.

### Bit 2 WRERRF: Write error flag
This bit is set when an unexpected write to the SAES_DINR register occurred. When set WRERRF bit has no impact on the SAES operations.
0: No error
1: Unexpected write to SAES_DINR register occurred during computation or data output phase.
The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.
The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

### Bit 1 RDERRF: Read error flag
This bit is set when an unexpected read to the SAES_DOUTR register occurred. When set RDERRF bit has no impact on the SAES operations.
0: No error
1: Unexpected read to SAES_DOUTR register occurred during computation or data input phase.
The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.
The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

### Bit 0 Reserved, must be kept at reset value.

### 34.8.3 SAES data input register (SAES_DINR)
Address offset: 0x008
Reset value: 0x0000 0000

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

### DIN[15:0]: Data input
A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes block of input data to the SAES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 16-bytes input buffer.
Reads return zero.
34.8.4 SAES data output register (SAES_DOUTR)
Address offset: 0x00C
Reset value: 0x0000 0000
Read when KMOD[1:0] is at 0x1 or 0x2 while MODE[1:0] is at 0x2 and EN is set triggers a read error.

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

Bits 31:0 DOUT[31:0]: Data output
This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte block of output data from the SAES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.
Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

34.8.5 SAES key register 0 (SAES_KEYR0)
Address offset: 0x010
Reset value: 0x0000 0000

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

Bits 31:0 KEY[31:0]: Cryptographic key, bits [31:0]
These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the SAES_CR register.
Writes to SAES_KEYRx registers are ignored when SAES is enabled (EN bit set). When KEYSEL[2:0] is different from 0 and KEYVALID is 0, writes to key registers are also ignored and they result in setting the KEIF bit of the SAES_ISR register.
With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.
34.8.6 SAES key register 1 (SAES_KEYR1)

Address offset: 0x014
Reset value: 0x0000 0000

Bits 31:0 KEY[63:32]: Cryptographic key, bits [63:32]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

34.8.7 SAES key register 2 (SAES_KEYR2)

Address offset: 0x018
Reset value: 0x0000 0000

Bits 31:0 KEY[95:64]: Cryptographic key, bits [95:64]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

34.8.8 SAES key register 3 (SAES_KEYR3)

Address offset: 0x01C
Reset value: 0x0000 0000

Bits 31:0 KEY[127:96]: Cryptographic key, bits [127:96]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.
### 34.8.9 SAES initialization vector register 0 (SAES_IVR0)

Address offset: 0x020  
Reset value: 0x0000 0000

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

Bits 31:0 IV[31:0]: Initialization vector input, bits [31:0]  
SAES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable).  
Write to this register is ignored when EN bit is set in SAES_SR register.

### 34.8.10 SAES initialization vector register 1 (SAES_IVR1)

Address offset: 0x024  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVI[63:48]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 IV[63:32]: Initialization vector input, bits [63:32]  
Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

### 34.8.11 SAES initialization vector register 2 (SAES_IVR2)

Address offset: 0x028  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVI[95:80]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 IV[95:64]: Initialization vector input, bits [95:64]  
Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.
34.8.12 SAES initialization vector register 3 (SAES_IVR3)

Address offset: 0x02C
Reset value: 0x0000 0000

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

IVI[127:112]

|   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 31| 30| 29| 28| 27| 26| 25| 24| 23| 22| 21| 20|
| rw| rw| rw| rw| rw| rw| rw| rw| rw| rw| rw| rw|

IVI[111:96]

Bits 31:0 IVI[127:96]: Initialization vector input, bits [127:96]
Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

34.8.13 SAES key register 4 (SAES_KEYR4)

Address offset: 0x030
Reset value: 0x0000 0000

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

KEY[159:144]

|   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 31| 30| 29| 28| 27| 26| 25| 24| 23| 22| 21| 20|
| w | w | w | w | w | w | w | w | w | w | w | w|

KEY[143:128]

Bits 31:0 KEY[159:128]: Cryptographic key, bits [159:128]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

34.8.14 SAES key register 5 (SAES_KEYR5)

Address offset: 0x034
Reset value: 0x0000 0000

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

KEY[191:176]

|   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 31| 30| 29| 28| 27| 26| 25| 24| 23| 22| 21| 20|
| w | w | w | w | w | w | w | w | w | w | w | w|

KEY[175:160]

Bits 31:0 KEY[191:160]: Cryptographic key, bits [191:160]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.
### 34.8.15 SAES key register 6 (SAES_KEYR6)

Address offset: 0x038  
Reset value: 0x0000 0000

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

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]  
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

### 34.8.16 SAES key register 7 (SAES_KEYR7)

Address offset: 0x03C  
Reset value: 0x0000 0000

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

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]  
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

### 34.8.17 SAES suspend registers (SAES_SUSPRx)

Address offset: 0x040 + x * 0x4, (x = 0 to 7)  
Reset value: 0x0000 0000

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

SUSP[31:16]

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

SUSP[15:0]
## 34.8.18 SAES interrupt enable register (SAES_IER)

Address offset: 0x300
Reset value: 0x0000 0000

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

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

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

**Bit 3** **RNGEIE**: RNG error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when RNGEIF (RNG error flag) is set.

0: Disabled (masked)
1: Enabled (not masked)

**Bit 2** **KEIE**: Key error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when KEIF (key error flag) is set.

0: Disabled (masked)
1: Enabled (not masked)

**Bit 1** **RWEIE**: Read or write error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when RWEIF (read and/or write error flag) is set.

0: Disabled (masked)
1: Enabled (not masked)

**Bit 0** **CCFIE**: Computation complete flag interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when CCF (computation complete flag) is set.

0: Disabled (masked)
1: Enabled (not masked)
34.8.19  SAES interrupt status register (SAES_ISR)

Address offset: 0x304

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-4</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

Bit 3 **RNGEIF**: RNG error interrupt flag

This read-only bit is set by hardware when an error is detected on RNG bus interface (for example bad entropy).

0: RNG bus is functional
1: Error detected on RNG bus interface (random seed fetching error)

The flag setting generates an interrupt if the RNGEIF bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register. The clear action triggers the reload of a new random number from the RNG peripheral.

Bit 2 **KEIF**: Key error interrupt flag

This read-only bit is set by hardware when the key information fails to load into key registers or when the key register use is forbidden.

0: No key error detected
1: Key information failed to load into key registers or the key register use is forbidden

The flag setting generates an interrupt if the KEIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

KEIF is raised upon any of the following events:
- SAES fails to load the DHUK (KEYSEL[2:0] = 0x1 or 0x4).
- SAES fails to load the BHK (KEYSEL[2:0] = 0x2 or 0x4).
- AES fails to load the key shared by SAES peripheral (KMOD[1:0] = 0x2).
- KEYVALID is set and either KEYPROT is set or KEYSEL[2:0] is other than 0x0. The security context of the application that loads the key (secure or non-secure) does not match the security attribute of the access to SAES_CR or SAES_DOUT. In this case, KEYVALID and EN bits are cleared.
- SAES_KEYRx register write does not respect the correct order. (For KEYSIZE cleared, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 register, or reverse. For KEYSIZE set, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 then SAES_KEYR4 then SAES_KEYR5 then SAES_KEYR6 then SAES_KEYR7, or reverse). KEIF must be cleared by the application software, otherwise KEYVALID cannot be set.
Bit 1 **RWEIF**: Read or write error interrupt flag
- This read-only bit is set by hardware when a RDERRF or a WRERRF error flag is set in the SAES_SR register.
  - 0: No read or write error detected
  - 1: Read or write error detected
- The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.
- The flag is cleared by setting the corresponding bit of the SAES_ICR register.
- The flags has no meaning when key derivation mode is selected.
  See the SAES_SR register for details.

Bit 0 **CCF**: Computation complete flag
- This flag indicates whether the computation is completed. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set.
  - 0: Not completed
  - 1: Completed
- The flag setting generates an interrupt if the CCFIE bit of the SAES_IER register is set.
- The flag is cleared by setting the corresponding bit of the SAES_ICR register.

### 34.8.20 SAES interrupt clear register (SAES_ICR)

Address offset: 0x308
Reset value: 0x0000 0000

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

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

Bit 3 **RNGEIF**: RNG error interrupt flag clear
- Application must set this bit to clear the RNGEIF status bit in SAES_ISR register.

Bit 2 **KEIF**: Key error interrupt flag clear
- Setting this bit clears the KEIF status bit of the SAES_ISR register.

Bit 1 **RWEIF**: Read or write error interrupt flag clear
- Setting this bit clears the RWEIF status bit of the SAES_ISR register, and clears both RDERRF and WRERRF flags in the SAES_SR register.

Bit 0 **CCF**: Computation complete flag clear
- Setting this bit clears the CCF status bit of the SAES_ISR register.
### Table 326. SAES register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>SAES_CR</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x008</td>
<td>SAES_DINR</td>
<td>DIN[31:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x00C</td>
<td>SAES_DOUTR</td>
<td>DOUT[31:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x010</td>
<td>SAES_KEYR0</td>
<td>KEY[31:0]</td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>SAES_KEYR1</td>
<td>KEY[83:32]</td>
<td></td>
</tr>
<tr>
<td>0x018</td>
<td>SAES_KEYR2</td>
<td>KEY[95:64]</td>
<td></td>
</tr>
<tr>
<td>0x01C</td>
<td>SAES_KEYR3</td>
<td>KEY[127:96]</td>
<td></td>
</tr>
<tr>
<td>0x020</td>
<td>SAES_IVR0</td>
<td>IVI[31:0]</td>
<td></td>
</tr>
<tr>
<td>0x024</td>
<td>SAES_IVR1</td>
<td>IVI[63:32]</td>
<td></td>
</tr>
<tr>
<td>0x028</td>
<td>SAES_IVR2</td>
<td>IVI[95:64]</td>
<td></td>
</tr>
<tr>
<td>0x02C</td>
<td>SAES_IVR3</td>
<td>IVI[127:96]</td>
<td></td>
</tr>
<tr>
<td>0x030</td>
<td>SAES_KEYR4</td>
<td>KEY[159:128]</td>
<td></td>
</tr>
<tr>
<td>0x034</td>
<td>SAES_KEYR5</td>
<td>KEY[191:160]</td>
<td></td>
</tr>
<tr>
<td>0x038</td>
<td>SAES_KEYR6</td>
<td>KEY[223:192]</td>
<td></td>
</tr>
<tr>
<td>0x03C</td>
<td>SAES_KEYR7</td>
<td>KEY[255:224]</td>
<td></td>
</tr>
</tbody>
</table>
Table 326. SAES register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x040  | SAES_SUSPR0   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x044  | SAES_SUSPR1   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x048  | SAES_SUSPR2   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x04C  | SAES_SUSPR3   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x050  | SAES_SUSPR4   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x054  | SAES_SUSPR5   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x058  | SAES_SUSPR6   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x05C  | SAES_SUSPR7   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x060-  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0FF  |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x300  | SAES_IER      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x304  | SAES_ISR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x308  | SAES_ICR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3 on page 111 for the register boundary addresses.
35 Hash processor (HASH)

35.1 Introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-2 family) and the HMAC (keyed-hash message authentication code) algorithm. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of any length less than $2^{64}$ bits (for SHA-1, SHA-224 and SHA-256) or less than $2^{128}$ bits (for SHA-384, SHA-512).

35.2 HASH main features

- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, Secure Hash Standard (SHA-1 and SHA-2 family)
  - Federal Information Processing Standards Publication FIPS PUB 186-4, Digital Signature Standard (DSS)
- Fast computation of SHA-1, SHA2-224, SHA2-256, SHA2-384, and SHA2-512
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
  - 98 clock cycles for processing one 1024-bit block of data using either SHA2-384 or SHA2-512 algorithm
  - Support for SHA-2 truncated outputs (SHA2-512/224, SHA2-512/256)
- Support for HMAC mode with all supported algorithm
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Supported word swapping format: bits, bytes, half-words and 32-bit words
- Single 32-bit, write-only, input register associated to an internal input FIFO, corresponding to a 64-byte block size (16 x 32 bits)
- Automatic padding to complete the input bit string to fit digest minimum block size
- AHB slave peripheral, accessible by 32-bit words only (else an AHB error is generated)
  - 8 x 32-bit words (H0 to H15) for output message digest
  - Automatic data flow control supporting direct memory access (DMA) using one channel.
  - Support for both single and fixed DMA burst transfers of four words.
  - Interruptible message digest computation, on a per-block basis
    - Re-loadable digest registers
    - Hashing computation suspend/resume mechanism, including DMA
35.3 HASH implementation

The devices have a single instance of HASH peripheral.

35.4 HASH functional description

35.4.1 HASH block diagram

*Figure 329* shows the block diagram of the hash processor.

35.4.2 HASH internal signals

*Table 327* describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hash_hclk</td>
<td>digital input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>hash_it</td>
<td>digital output</td>
<td>Hash processor global interrupt request</td>
</tr>
<tr>
<td>hash_dma</td>
<td>digital input/output</td>
<td>DMA burst request/ acknowledge</td>
</tr>
</tbody>
</table>
35.4.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard.

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message is presented on the input, the HASH processing core produces a fixed-length output string called a message digest (see Table 328).

Table 328. Information on supported hash algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Message digest size (in bits)</th>
<th>Block size (in bytes)(1)</th>
<th>Message length</th>
<th>Bit string message</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1</td>
<td>160</td>
<td>64</td>
<td>&lt; 2^{64} bits</td>
<td>Yes</td>
</tr>
<tr>
<td>SHA2-224</td>
<td>224</td>
<td>64</td>
<td>&lt; 2^{128} bits</td>
<td>Yes</td>
</tr>
<tr>
<td>SHA2-256</td>
<td>256</td>
<td>128</td>
<td>&lt; 2^{128} bits</td>
<td>Yes</td>
</tr>
<tr>
<td>SHA2-384</td>
<td>384</td>
<td>128</td>
<td>&lt; 2^{128} bits</td>
<td>Yes</td>
</tr>
<tr>
<td>SHA2-512</td>
<td>512(2)</td>
<td>128</td>
<td>&lt; 2^{128} bits</td>
<td>Yes</td>
</tr>
</tbody>
</table>

1. Block size = (NBWE-1) * 4 bytes. NBWE[4:0] bitfield can be read from the HASH_SR register after ALGO and INIT bits are written in the HASH_CR register.
2. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes).

The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process since the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” by NIST because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest (SHA-1 does not qualify as secure since February 2017). Any change to a message in transit results, with very high probability, in a different message digest, and the signature fails to verify.

35.4.4 Message data feeding

The message (or data file) to be processed by the HASH must be considered as a bit string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string “abc” with a bit string representation of “01100001 01100010 01100011” is represented by a 32-bit word 0x00636261, and 8-bit words 0x61626300.

Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO each time the register is written with new data. Hence HASH_DIN and the FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).

In accordance to the kind of data to be processed (for example byte swapping when data are ASCII text stream) there must be a bit, byte, half-word or no swapping operation to be
performed on data from the input FIFO before entering the little-endian hash processing core. Figure 330 shows how the hash processing core 32-bit data block M0...31 is constructed from one 32-bit words popped into input FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR)

HASH_DIN data endianness when bit swapping is disabled (DATATYPE = 00) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

Figure 330. Message data swapping feature
35.4.5 Message digest computing

The hash processor sequentially processes several blocks when computing the message digest. Block sizes can be found in Table 328: Information on supported hash algorithms.

Each time the DMA or the CPU writes a block to the hash processor, the HASH automatically starts computing the message digest. This operation is known as partial digest computation.

As described in Section 35.4.4: Message data feeding, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the input FIFO. In order to perform the hash computation on this data the application must follow below sequence.

1. Initialize the hash processor using the HASH_CR register:
   - Write to the HASH_CR register to select the right algorithm using the ALGO bitfield, and set the INIT bit (other bits are kept at zero). Then read the NBWE bitfield from the HASH_SR register to deduce the algorithm block size, which equals \((\text{NBWE} - 1) \times 4\). This step is not required if the block size is already known (see Table 328 for details).
   - Select the right algorithm using the ALGO[3:0] field. If needed, program the correct swapping operation on the message input words using the DATATYPE[1:0] bitfield.
   - When HMAC mode is required, set the MODE bit as well as the LKEY bit if the HMAC key size is greater than the known block size of the algorithm (otherwise keep LKEY cleared). Refer to Section 35.4.7: HMAC operation for details.
   - Update NBLW[4:0] in the HASH_STR register to define the number of valid bits in the last word of the message if it is different from 32 bits. NBLW information are used to correctly perform the automatic message padding before the final message digest computation.

2. Complete the initialization by setting the INIT bit in HASH_CR register. Also set the DMAE bit if data are transferred via DMA.

Caution: When programming step 2, it is important that the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW) are set up before or at the same time.

3. Start filling data by writing to the HASH_DIN register, unless data are automatically transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the input FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   - Data are filled by software:
     Partial digest computations are triggered each time the application writes the first word of the next block, the block size being defined by NBWE bits in HASH_SR. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.
     The final digest computation is triggered when the last block is entered and the software sets the DCAL bit. If the message length is not an exact multiple of the
block size, the NBLW field in HASH_STR register must be written prior to writing DCAL bit (see Section 35.4.6 for details).

- Data are filled as a single DMA transfer (MDMAT = 0):
  Partial digest computations are triggered automatically each time the FIFO is full.
  The final digest computation is triggered automatically when the last block has
  been transferred to the HASH_DIN register by DMA (DCAL bit is set by hardware).
  If the message length is not an exact multiple of the block size, the NBLW field in
  HASH_STR register must be written prior to enabling the DMA (see
  Section 35.4.6 for details).

- Data are filled using multiple DMA transfers (MDMAT = 1):
  Partial digest computations are triggered as for single DMA transfers (refer to the
  above description). However the final digest computation is not triggered
  automatically when the last block has been transferred by DMA to the HASH_DIN
  register (DCAL bit is not set by hardware). It enables the hash processor to
  receive a new DMA transfer as part of this digest computation. To launch the final
  digest computation, the software must clear MDMAT bit before the last DMA
  transfer in order to trigger the final digest computation as it is done for single DMA
  transfers.

4. Once the digest calculation is completed (DCIS = 1), the resulting digest can be read
   from the output registers, as described in Table 329.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Valid output registers</th>
<th>Most significant bit</th>
<th>Digest size (in bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1</td>
<td>HASH_H0 to HASH_H4</td>
<td>HASH_H0[31]</td>
<td>160</td>
</tr>
<tr>
<td>SHA2-224</td>
<td>HASH_H0 to HASH_H6</td>
<td>HASH_H0[31]</td>
<td>224</td>
</tr>
<tr>
<td>SHA2-256</td>
<td>HASH_H0 to HASH_H7</td>
<td>HASH_H0[31]</td>
<td>256</td>
</tr>
<tr>
<td>SHA2-384</td>
<td>HASH_H0 to HASH_H11</td>
<td>HASH_H0[31]</td>
<td>384</td>
</tr>
<tr>
<td>SHA2-512</td>
<td>HASH_H0 to HASH_H15</td>
<td></td>
<td>512(1)</td>
</tr>
</tbody>
</table>

1. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes)

For more information about HMAC detailed instructions, refer to Section 35.4.7: HMAC operation.

### 35.4.6 Message padding

#### Overview

When computing a condensed representation of a message, the process of feeding data
into the hash processor (with automatic partial digest computation every block size transfer)
loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written
to the hash processor may have a valid number of bits between 1 and 32. This number of
valid bits in the last word, NBLW, has to be written to the HASH_STR register, so that
message padding is correctly performed before the final message digest computation.
Padding processing

Detailed padding sequences, with DMA enabled or disabled, are described in Section 35.4.5: Message digest computing.

Padding example

As specified by Federal Information Processing Standards PUB 180-4, message padding consists in appending a “1” followed by \( k \) “0”s, itself followed by a 64-bit integer that is equal to the length \( L \) in bits of the message. These three padding operations generate a padded message of length \( L + 1 + k + 64 \), which by construction is a multiple of 512 bits.

For the hash processor, the “1” is added to the last word written to the HASH_DIN register at the bit position defined by the NBLW bitfield, and the remaining upper bits are cleared (“0”s).

Example from FIPS PUB180-4

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length \( L = 24 \):

```
byte 0  byte 1  byte 2  byte 3
01100001 01100010 01100011 UUUUUUU
<--- 1st word written to HASH_DIN -->
```

NBLW has to be loaded with the value 24: a “1” is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

```
01100001 01100010 01100011 IUUUUUUU
```

Since \( L = 24 \), the number of bits in the above bit string is 25, and 423 “0” bits are appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000
```

The message length value, \( L \), in two-word format (that is 00000000 00000018) is appended. Hence the final padded message in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
```

If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE = 10 in HASH_CR), the above message has to be entered using following sequence:

1. \texttt{0xUU636261} is written to the HASH_DIN register (where ‘U’ means don’t care)
2. \texttt{0x18} is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits)
3. \texttt{0x10} is written to the HASH_STR register to start the message padding (described above) and then perform the digest computation.
4. The hash computing is complete with the message digest available in the HASH_HRx registers (x = 0...4) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

   \begin{align*}
   \text{HASH_HR0} &= \texttt{0xA9993E36} \\
   \text{HASH_HR1} &= \texttt{0x4706816A} \\
   \text{HASH_HR2} &= \texttt{0xBA3E2571} \\
   \text{HASH_HR3} &= \texttt{0x7850C26C} \\
   \text{HASH_HR4} &= \texttt{0x9CD0D89D}
   \end{align*}

### 35.4.7 HMAC operation

#### Overview

As specified by Internet Engineering Task Force RFC2104 and NIST FIPS PUB 198-1, the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

\[
\text{HMAC(}\text{message}\text{)} = \text{Hash}(\text{Key} \ | \ \text{pad}) \text{ XOR opad} \ | \\
\text{Hash}(\text{Key} \ | \ \text{pad}) \text{ XOR ipad} \ | \ \text{message})
\]

where:

- \text{opad} = [0x5C]_n (outer pad) and \text{ipad} = [0x36]_n (inner pad)
- \([X]_n\) represents a repetition of \(X\) \(n\) times, where \(n\) equal to the byte size of the underlying hash function data block (\(n = 64\) when block size is \(512\) bits).
- \text{pad} is a sequence of zeroes needed to extend the key to the length \(n\) defined above. If the key length is greater than \(n\), the application must first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- \(\mid\) represents the concatenation operator.

\textbf{Note:} \textit{HMAC mode of the hash processor can be used with all supported algorithms.}

#### HMAC processing

Four different steps are required to compute the HMAC:

1. The software sets the INIT bit, with the MODE bit set and the ALGO bits selecting the desired algorithm. The LKEY bit must also be set if the key being used is longer than the block size. In this case, as required by HMAC specifications, the hash processor uses the hash of the key instead of the real key.
2. The software provides the key to be used for the inner hash function, using the same mechanism as the message string loading, that is by writing the key data into
HASH_DIN register and then completing the transfer by setting DCAL bit and the correct NBLW to HASH_STR register.

3. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write the message string to HASH_DIN. When the last word of the last block is entered and the software sets DCAL bit in HASH_STR register, the NBLW bitfield must be programmed at the same time to a value different from zero if the message length is not an exact multiple of the block size. Note that the DMA can also be used to feed the message string, as described in Section 35.4.5: Message digest computing.

4. Once the processor is ready again (DINIS = 1 in HASH_SR), the software provides the key to be used for the outer hash function, writing the key data into HASH_DIN register, and then completing the transfer by setting DCAL bit and programming the correct NBLW to HASH_STR register. The HMAC result can be found in the valid output registers (HASH_HRx) as soon as DCIS bit is set.

Note: The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in Section 35.6: HASH processing time.

Endianness management details can be found in Section 35.4.4: Message data feeding.

HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO = 00 and MODE = 1 in HASH_CR) as specified by NIST. SHA-1 block size is 64 bytes.

Let us assume that the original message is the ASCII binary-coded form of "Sample message for keylen = blocklen", of length L = 34 bytes. If the HASH is programmed in no swapping mode (DATATYPE = 00 in HASH_CR), the following data must be loaded sequentially into HASH_DIN register:

1. **Inner hash key** input (length = 64, i.e. no padding), specified by NIST. As key length = 64, LKEY bit is cleared in HASH_CR register
   
   00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617
   18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F
   30313233 34353637 38393A3B 3C3D3E3F

2. **Message** input (length = 34, i.e. padding required). HASH_STR must be set to 0x20 to start message padding and inner hash computation (see 'U' as don’t care)
   
   53616D70 6C65206D 65737361 67652066 6F722065
   73696C65 53616D70 6C65206D 65737361 67652066 6F722065
   73696C65 53616D70 6C65206D 65737361 67652066 6F722065
   73696C65

3. **Outer hash key** input (length = 64, i.e. no padding). A key identical to the inner hash key is entered here.

4. **Final outer hash computing** is then performed by the HASH. The HMAC-SHA1 digest result is available in the HASH_HRx registers, as shown below:

   HASH_HR0 = 0xF5FD596EE
   HASH_HR1 = 0x78D5555C
   HASH_HR2 = 0x8FF4E72D
   HASH_HR3 = 0x266DFD19
   HASH_HR4 = 0x2366DA29
35.4.8 HASH suspend/resume operations

Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in Figure 331.

Figure 331. HASH suspend/resume mechanism

To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described hereafter.
Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing.

To suspend the processing of a message, proceed as follows after writing the number of words defined in NBWE:

1. In Polling mode, wait for BUSY = 0 then poll if the DINIS status bit is set.
   In Interrupt mode, implement the next step in DINIS interrupt handler (recommended).
2. Store the contents of the following registers into memory:
   - HASH_IMR
   - HASH_STR
   - HASH_CR
   - HASH_CSR0 to HASH_CSR37, when SHA-1 or SHA2-256 is selected, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing
   - HASH_CSR0 to HASH_CSR90, when SHA2-384 or SHA2-512 (truncated or not) is selected, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.

To resume the processing of a message, proceed as follows:

1. Write the following registers with the values saved in memory: HASH_IMR, HASH_STR, HASH_CR.
2. Initialize the hash processor by setting the INIT bit in the HASH_CR register
3. Write the HASH_CSRx registers with the values saved in memory.
4. Restart the processing from the point where it has been interrupted.

Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is recommended to suspend and then restore a secure digest computing as described below.

In this sequence the DMA channel allocated to the hash peripheral remains allocated to the processing of message 1 (see Figure 331).

To suspend the processing of a message using DMA, proceed as follows:

1. Clear the DMAE bit to disable the DMA interface. The hash peripheral automatically fetches enough data via the DMA to complete the current burst transfer.
2. Wait until the last DMA transfer is complete (DMAS = 0 in HASH_SR).
3. Disable the DMA channel.
4. In Polling or Interrupt mode (recommended), wait until the hash processor is ready (no block is being processed), that is wait for DINIS = 1 in HASH_SR. If DCIS is also set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 5.
5. Save HASH_IMR, HASH_STR and HASH_CR registers. Also save a number of HASH_CSRx registers depending on the SHA algorithm that is used:
   - When SHA-1 or SHA2-256 is selected, save HASH_CSR0 to HASH_CSR37, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing.
   - When SHA2-384 or SHA2-512 is selected, save HASH_CSR0 to HASH_CSR90, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.
To resume the processing of a message using DMA, proceed as follows:

1. Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again.
2. Program the values saved in memory to HASH_IMR, HASH_STR and HASH_CR registers.
3. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
4. Program the values saved in memory to the HASH_CSRx registers.
5. Restart the processing from the point where it was interrupted by setting the DMAE bit.

35.4.9 HASH DMA interface

The HASH supports both single and fixed DMA burst transfers of four words.

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH_CR register. When this bit is set, the HASH initiates a DMA request each time a block has to be written to the HASH_DIN register.

Once four 32-bit words have been received, the HASH automatically triggers a new request to the DMA. For more information refer to Section 35.4.5: Message digest computing.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that is copied into HASH_DIN register. This is done by writing in HASH_STR register the following value:

\[ NBLW = \text{Len(Message)} \mod 32 \]

where \( x \mod 32 \) gives the remainder of \( x \) divided by 32.

The DMAS bit of the HASH_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared and no DMA transfer is ongoing.

**Note:** No interrupt is associated to DMAS bit.

*When MDMAT is set, the size of the transfer must be a multiple of four words.*

35.4.10 HASH error management

No error flags are generated by the hash processor.

35.5 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal the following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal (hash_it), which is in turn connected to the device interrupt controller. Each interrupt source can individually be enabled or disabled by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit enables the interrupt.

The status of each maskable interrupt sources can be read from the HASH_SR register.

*Table 330* gives a summary of the available features.
35.6 HASH processing time

Table 331 summarizes the time required to process an intermediate block for each mode of operation.

<table>
<thead>
<tr>
<th>Mode of operation</th>
<th>Block size (in bytes)</th>
<th>FIFO load</th>
<th>Computation phase</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1</td>
<td>64</td>
<td>16</td>
<td>66</td>
<td>82</td>
</tr>
<tr>
<td>SHA2-224</td>
<td></td>
<td>16</td>
<td>50</td>
<td>66</td>
</tr>
<tr>
<td>SHA2-256</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHA2-384</td>
<td>128</td>
<td>32</td>
<td>66</td>
<td>98</td>
</tr>
<tr>
<td>SHA2-512(2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Add the time required to load the block into the processor.
2. SHA2-512 includes SHA2-512/224 and SHA2-512/256 modes.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:
- 1 to 2.5 for a hash message
- ~2.5 for an HMAC input-key
- 1 to 2.5 for an HMAC message
- ~2.5 for an HMAC output key in case of a short key
- 3.5 to 5 for an HMAC output key in case of a long key

35.7 HASH registers

The hash core is associated with several control and status registers and several message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB error is generated.

35.7.1 HASH control register (HASH_CR)

Address offset: 0x00
Reset value: 0x0000 0000
Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **ALGO[3:0]: Algorithm selection**

These bits select the hash algorithm:

- 0000: SHA-1
- 0001: reserved
- 0010: SHA2-224
- 0011: SHA2-256
- 1100: SHA2-384
- 1101: SHA2-512/224
- 1110: SHA2-512/256
- 1111: SHA2-512

This selection is only taken into account when the INIT bit is set. Changing this bitfield during a computation has no effect.

When the ALGO bitfield is updated and INIT bit is set, NBWE in HASH_SR is automatically updated to 0x11.

Bit 16 **LKEY**: Long key selection

The application must set this bit if the HMAC key is greater than the block size corresponding to the hash algorithm (see Table 328: Information on supported hash algorithms for details). For example the block size is 64 bytes for SHA2-256.

- 0: HMAC key is shorter or equal to the block size (short key). The actual key value written in HASH_DIN is used during the HMAC computation.
- 1: HMAC key is longer than the block size (long key). The hash of the key is used instead of the real key during the HMAC computation.

This selection is only taken into account when the INIT and MODE bits are set (HMAC mode selected). Changing this bit during a computation has no effect.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **MDMAT**: Multiple DMA transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

- 0: DCAL is automatically set at the end of a DMA transfer.
- 1: DCAL is not automatically set at the end of a DMA transfer.

Bit 12 **DINNE**: DIN not empty

Refer to DINNE bit of HASH_SR for a description of DINNE bit.

This bit is read-only.

Bits 11:8 **NBW[3:0]: Number of words already pushed**


This bit is read-only.

Bit 7 Reserved, must be kept at reset value.
Bit 6 **MODE:** Mode selection
This bit selects the normal or the keyed HMAC mode for the selected algorithm:
0: Hash mode selected
1: HMAC mode selected. LKEY bit must be set if the key being used is longer than the algorithm block size.
This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

Bits 5:4 **DATATYPE[1:0]:** Data type selection
This bitfield defines the format of the data entered into the HASH_DIN register:
00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.
01: 16-bit data or half-word. The data written into HASH_DIN are considered as two half-words, and are swapped before being used by the HASH processing.
10: 8-bit data or bytes. The data written into HASH_DIN are considered as four bytes, and are swapped before being used by the HASH processing.
11: bit data or bit string. The data written into HASH_DIN are considered as 32 bits (1st bit of the string at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).

Bit 3 **DMAE:** DMA enable
0: DMA transfers disabled
1: DMA transfers enabled. A DMA request is sent as soon as the hash core is ready to receive data.
After this bit is set, it is cleared by hardware while the last data of the message is written into the hash processor.
Clearing this bit while a DMA transfer is ongoing does not abort the current transfer. Instead, the DMA interface of the HASH remains internally enabled until the transfer is complete or INIT is set.
Setting INIT bit does not clear DMAE bit.

Bit 2 **INIT:** Initialize message digest calculation
Setting this bit resets the hash processor core, so that the HASH is ready to compute the message digest of a new message.
Clearing this bit has no effect. Reading this bit always returns 0.

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

### 35.7.2 HASH data input register (HASH_DIN)

**Address offset:** 0x04

**Reset value:** 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks defined by the hash algorithm (see Table 328: Information on supported hash algorithms for details). For example the block size is 64 bytes for SHA2-256.

When the HASH_DIN register is programmed, the value presented on the AHB bus is ‘pushed’ into the hash core and the register takes the new value presented on the AHB bus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH_CR register.
When a complete block has been written to the HASH_DIN register, an intermediate digest calculation is launched:
- by writing first data of the next block into the HASH_DIN register, if the DMA is not used
- automatically, if the DMA is used

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched by setting the DCAL bit in the HASH_STR register (final digest calculation). This operation is automatic if the DMA is used and MDMAT bit is cleared.

Reading the HASH_DIN register returns zeros.

Note: When the HASH is busy, a write access to the HASH_DIN register might stall the AHB bus if the digest calculation (intermediate or final) is not complete.

35.7.3 HASH start register (HASH_STR)

Address offset: 0x08

Reset value: 0x0000 0000

The HASH_STR register has two functions:
- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH_DIN register)
- It is used to start the processing of the last block in the message by setting the DCAL bit.
Bits 31:9  Reserved, must be kept at reset value.

Bit 8  **DCAL**: Digest calculation
Setting this bit starts the message padding using the previously written value of NBLW, and starts the calculation of the final message digest with all the data words written to the input FIFO since the INIT bit was last set.
Reading this bit returns 0.

Bits 7:5  Reserved, must be kept at reset value.

Bits 4:0  **NBLW[4:0]**: Number of valid bits in the last word
When the last word of the message bit string is written to HASH_DIN register, the hash processor takes only the valid bits, specified as below, after internal data swapping:
0x00: All the 32 bits of the last data written are valid message bits, that is M[31:0]
0x01: Only one bit of the last data written (after swapping) is valid, that is M[0]
0x02: Only two bits of the last data written (after swapping) are valid, that is M[1:0]
0x03: Only three bits of the last data written (after swapping) are valid that is M[2:0]
...
0x1F: Only 31 bits of the last data written (after swapping) are valid that is M[30:0]
The above mechanism is valid only if DCAL = 0. If NBLW bits are written while DCAL is set, the NBLW bitfield remains unchanged. In other words it is not possible to configure NBLW and set DCAL at the same time.
Reading NBLW bits returns the last value written to NBLW.

### 35.7.4 HASH digest registers

These registers contain the message digest result defined as follows:
- HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 registers return the SHA-1 digest result.
- HASH_HR0 to HASH_HR6 registers return the SHA2-224, SHA2-512/224 digest result.
- HASH_HR0 to HASH_HR7 registers return the SHA2-256, SHA2-512/256 digest result.
- HASH_HR0 to HASH_HR11 registers return the SHA2-384 digest result.
- HASH_HR0 to HASH_HR15 registers return the SHA2-512 digest result

In all cases, the digest most significant bit is stored in HASH_HR0[31], and unused HASH_HRx registers reads as zero.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (DCIS bit equals 0), then the read operation returns zeros.

**Note:** When starting a digest computation for a new message (by setting the INIT bit), HASH_HRx registers are forced to their reset values.

**HASH_HR0 to HASH_HR4 registers can be accessed through two different addresses (register aliasing).**
**HASH aliased digest register x (HASH_HRAx)**

Address offset: 0x0C + x * 0x4, (x = 0 to 4)
Reset value: 0x0000 0000

The content of the HASH_HRAx registers is identical to the one of the HASH_HRx registers located at address offset 0x310.

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

| r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  |

Bits 31:0 **Hx[31:0]:** Hash data x  
Refer to [Section 35.7.4: HASH digest registers](#) introduction.

**HASH digest register x (HASH_HRx)**

Address offset: 0x310 + x * 0x4, (x = 0 to 4)
Reset value: 0x0000 0000

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

| r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  |

Bits 31:0 **Hx[31:0]:** Hash data x  
Refer to [Section 35.7.4: HASH digest registers](#) introduction.

**HASH supplementary digest register x (HASH_HRx)**

Address offset: 0x310 + x * 0x4, (x = 5 to 15)
Reset value: 0x0000 0000

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

| r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  |

Bits 31:0 **Hx[31:0]:** Hash data x  
Refer to [Section 35.7.4: HASH digest registers](#) introduction.
### 35.7.5 HASH interrupt enable register (HASH_IMR)

Address offset: 0x20  
Reset value: 0x0000 0000

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

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

Bit 1 **DCIE**: Digest calculation completion interrupt enable  
0: Digest calculation completion interrupt disabled  
1: Digest calculation completion interrupt enabled.

Bit 0 **DINIE**: Data input interrupt enable  
0: Data input interrupt disabled  
1: Data input interrupt enabled

### 35.7.6 HASH status register (HASH_SR)

Address offset: 0x24  
Reset value: 0x0011 0001

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

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **NBWE[4:0]**: Number of words expected  
This bitfield reflects the number of words in the message that must be pushed into the FIFO to trigger a partial computation. NBWE is decremented by 1 when a write access is performed to the HASH_DIN register. NBWE is set to the expected block size +1 in words (0x11) when INIT bit is set in HASH_CR. It is set to the expected block size (0x10) when the partial digest calculation ends.

Bit 15 **DINNE**: DIN not empty  
This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is set.  
0: No data are present in the data input buffer  
1: The input buffer contains at least one word of data

Bit 14 Reserved, must be kept at reset value.
35.7.7 HASH context swap registers

These registers contain the complete internal register states of the hash processor. They are useful when a suspend/resume operation has to be performed because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task. When the hash computation is complete, the saved context can be read from memory and written back into the HASH_CSRx registers.

HASH_CSRx registers can be read only when DINIS equals to 1, otherwise zeros are returned.
HASH context swap register x (HASH_CSRx)

Address offset: 0x0F8 + x * 0x4, (x = 0 to 102)
Reset value: 0x0022 0002 (HASH_CSR0)
Reset value: 0x0020 0000 (HASH_CSR2)
Reset value: 0x0000 0000 (others)

35.7.8 HASH register map

Table 332. HASH1 register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>HASH_CR</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x04</td>
<td>HASH/Register</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x08</td>
<td>HASH/STR</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0C</td>
<td>HASH/HR0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x10</td>
<td>HASH/HR1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x14</td>
<td>HASH/HR2</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x18</td>
<td>HASH/HR3</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x1C</td>
<td>HASH/HR4</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x20</td>
<td>HASH/IMR</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x24</td>
<td>HASH/SR</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x28</td>
<td>HASH/CSR0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x2C</td>
<td>HASH/CSR1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x30</td>
<td>HASH/CSR2</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x34</td>
<td>HASH/CSR3</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Bits 31:0 CSx[31:0]: Context swap x
Refer to Section 35.7.7: HASH context swap registers introduction.
### Table 332. HASH1 register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x290</td>
<td>HASH_CSR102</td>
<td>CSR1[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x310</td>
<td>HASH_HR0</td>
<td>H0[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x314</td>
<td>HASH_HR1</td>
<td>H1[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x318</td>
<td>HASH_HR2</td>
<td>H2[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x31C</td>
<td>HASH_HR3</td>
<td>H3[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x320</td>
<td>HASH_HR4</td>
<td>H4[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x324</td>
<td>HASH_HR5</td>
<td>H5[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x328</td>
<td>HASH_HR6</td>
<td>H6[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x32C</td>
<td>HASH_HR7</td>
<td>H7[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x33C</td>
<td>HASH_HR15</td>
<td>R15[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 111 for the register boundary addresses.
36 Public key accelerator (PKA)

36.1 Introduction

PKA (public key accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over GF(p) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

For a given operation, all needed computations are performed within the accelerator, so no further hardware/software elaboration is needed to process the inputs or the outputs.

When manipulating secrets, the PKA incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3.

36.2 PKA main features

- Acceleration of RSA, DH and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA chinese remainder theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check, complete addition, double base ladder, projective to affine
  - ECDSA signature generation and verification
- Capability to handle operands up to 4160 bits for RSA/DH and 640 bits for ECC
- When manipulating secrets: protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3
  - Applicable to modular exponentiation, ECC scalar multiplication and ECDSA signature generation
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication
- Built-in Montgomery domain inward and outward transformations
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
36.3 PKA functional description

36.3.1 PKA block diagram

Figure 332. PKA block diagram

36.3.2 PKA internal signals

Table 333 lists the internal signals available at the PKA level, not necessarily available on product bonding pads.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pka_hclk</td>
<td>Digital input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>pka_it</td>
<td>Digital output</td>
<td>Public key accelerator IP global interrupt request</td>
</tr>
<tr>
<td>pka_itamp_out</td>
<td>Digital output</td>
<td>PKA internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected fault occurs while PKA manipulates secrets, or when the programmed input point is not found on the input curve (ECDSA signature and ECC scalar multiplication only). This signal is asserted as soon as a fault is detected. When asserted, read access to PKA registers are reset to 0 and writes are ignored. The signal is de-asserted when PKA memory is cleared.</td>
</tr>
</tbody>
</table>

36.3.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. When the PKA peripheral reset signal is released PKA_RAM is cleared automatically, taking 667 clock cycles. During this time the setting of bit EN in PKA_CR register is ignored.

According to the security policy applied to the device, PKA RAM can also be reset following a tamper event. Refer to *Tamper detection and response* in the System security section (if applicable to this product).
36.3.4 PKA public key acceleration

Overview
Public key accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 4160 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation $y^2 = x^3 + ax + b \pmod{p}$. More information can be found in Section 36.5.1: Supported elliptic curves.

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

A memory of 5336 bytes (667 words of 64 bits) called PKA RAM is used to provide initial data to the PKA, and to hold the results after computation is completed. Access is done through the PKA AHB interface.

PKA operating modes
The list of operations the PKA can perform is detailed in Table 334 and Table 335, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

<table>
<thead>
<tr>
<th>PKA_CR.MODE[5:0]</th>
<th>Performed operation</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>000001 Montgomery parameter computation R2 mod n</td>
<td>Section 36.4.2</td>
</tr>
<tr>
<td>0x0E</td>
<td>001110 Modular addition (A+B) mod n</td>
<td>Section 36.4.3</td>
</tr>
<tr>
<td>0x0F</td>
<td>001111 Modular subtraction (A-B) mod n</td>
<td>Section 36.4.4</td>
</tr>
<tr>
<td>0x10</td>
<td>010000 Montgomery multiplication (AxB) mod n</td>
<td>Section 36.4.5</td>
</tr>
<tr>
<td>0x00</td>
<td>000000 Modular exponentiation A^e mod n</td>
<td>Section 36.4.6</td>
</tr>
<tr>
<td>0x02</td>
<td>000010 Modular exponentiation A^e mod n (fast mode)</td>
<td>Section 36.4.6</td>
</tr>
<tr>
<td>0x03</td>
<td>000011 Modular exponentiation A^e mod n (protected)</td>
<td>Section 36.4.6</td>
</tr>
<tr>
<td>0x08</td>
<td>001000 Modular inversion A^{-1} mod n</td>
<td>Section 36.4.7</td>
</tr>
<tr>
<td>0x0D</td>
<td>001100 Modular reduction A mod n</td>
<td>Section 36.4.8</td>
</tr>
<tr>
<td>0x09</td>
<td>001001 Arithmetic addition A+B</td>
<td>Section 36.4.9</td>
</tr>
<tr>
<td>0x0A</td>
<td>001010 Arithmetic subtraction A-B</td>
<td>Section 36.4.10</td>
</tr>
<tr>
<td>0x0B</td>
<td>001111 Arithmetic multiplication AxB</td>
<td>Section 36.4.11</td>
</tr>
<tr>
<td>0x0C</td>
<td>001100 Arithmetic comparison (A=B, A&gt;B, A&lt;B)</td>
<td>Section 36.4.12</td>
</tr>
<tr>
<td>0x07</td>
<td>000111 RSA CRT exponentiation</td>
<td>Section 36.4.13</td>
</tr>
</tbody>
</table>
Each of these operating modes has an associated code that has to be written to the MODE field in the PKA_CR register. If the application selects any value that is not documented below the write to MODE bitfield is ignored, and an operation error (OPERRF) is triggered. When this happens, a new operation must be selected after the error is cleared.

Some operations in Table 334 and Table 335 are indicated as protected. Those operations are used when manipulating secret keys (modular exponentiation for RSA decryption, scalar multiplication and signature for ECC). Those secrets (protected against side channel attacks) are automatically erased from PKA RAM at the end of the protected operations (BUSY goes low). They are also protected against side channel attacks.

**Caution:** For security reason it is very important to select protected modular exponentiation (MODE = 0x3) when performing RSA decryption.

### Montgomery space and fast mode operations

For efficiency reason the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming the application can decide to use a faster mode of operation, during which the precomputed Montgomery parameter is supplied before starting the operation. Performance improvement is detailed in Section 36.5.2: Computation times.

The only operation using fast mode is modular exponentiation (MODE = 0x02).

### 36.3.5 Typical applications for PKA

**Introduction**

The PKA can be used to accelerate a number of public key cryptographic functions. In particular:

- RSA encryption and decryption
- RSA key finalization
- CRT-RSA decryption
- DSA and ECDSA signature generation and verification
- DH and ECDH key agreement
Specifications of the above functions are given in following publications:

- FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
- PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories

The principles of the main functions are described in this section, for a more detailed description refer to the above cited documents.

**RSA key pair**

For the following RSA operations a public key and a private key information are defined as below:

- Alice transmits her public key \((n, e)\) to Bob. Numbers \(n\) and \(e\) are very large positive integers.
- Alice keeps secret her private key \(d\), also a very large positive integer. Alternatively this private key can also be represented by a quintuple \((p, q, dp, dq, qInv)\).

For more information on the above representations refer to the RSA specification.

**RSA encryption/decryption principle**

As recommended by the PKCS#1 specification, Bob, to encrypt message \(M\) using Alice’s public key \((n, e)\) must go through the following steps:

1. Compute the encoded message \(EM = ENCODE(M)\), where ENCODE is an encoding method.
2. Turn \(EM\) into an integer \(m\), with \(0 \leq m < n\) and \((m, n)\) being coprimes.
3. Compute ciphertext \(c = m^e \mod n\).
4. Convert the integer \(c\) into a string ciphertext \(C\).

Alice, to decrypt ciphertext \(C\) using her private key \(d\), follows the steps indicated below:

1. Convert the ciphertext \(C\) to an integer ciphertext representative \(c\).
2. If necessary, retrieve the prime factors \((p, q)\) using \((n, e, d)\) information, then compute \(\phi = (p - 1) \times (q - 1)\). Refer to NIST SP800-56B Appendix C for details.
3. Recover plaintext \(m = c^d \mod n = (m^e)^d \mod n\). If the private key is the quintuple \((p, q, dp, dq, qInv)\), then plaintext \(m\) is obtained by performing the operations:
   a) \(m_1 = c^{dp} \mod p\)
   b) \(m_2 = c^{dq} \mod q\)
   c) \(h = qInv \times (m_1 - m_2) \mod p\)
   d) \(m = m_2 + h \times q\)
4. Convert the integer message representative \(m\) to an encoded message \(EM\).
5. Recover message \(M = DECODE(EM)\), where DECODE is a decoding method.

Above operations can be accelerated by PKA using Modular exponentiation \(A^e \mod n\) if the private key is \(d\), or RSA CRT exponentiation if the private key is the quintuple \((p, q, dp, dq, qInv)\).

*Note:* The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.
Note: For the decryption process protected version of modular exponentiation (MODE = 0x3) is strongly recommended for security reason. For encryption process MODE = 0x3 cannot be used, as it requires the knowledge of the private key.

Elliptic curve selection

For following ECC operations curve parameters are defined as below:
- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob). Supported curves parameters are summarized in Section 36.5.1: Supported elliptic curves.
- G is the chosen elliptic curve base point (also known as generator), with a large prime order n (i.e. \( n \times G = \text{identity element } O \)).

ECDSA message signature generation

ECDSA (elliptic curve digital signature algorithm) signature generation function principle is the following: Alice, to sign a message \( m \) using her private key integer \( d_A \), goes through the following steps.
1. Calculate \( e = \text{HASH}(m) \), where \( \text{HASH} \) is a cryptographic hash function.
2. Let \( z \) be the \( L_n \) leftmost bits of \( e \), where \( L_n \) is the bit length of the group order \( n \).
3. Select a cryptographically secure random integer \( k \) where \( 0 < k < n \).
4. Calculate the curve point \( (x_1, y_1) = k \times G \).
5. Calculate \( r = x_1 \mod n \). If \( r = 0 \) go back to step 3.
6. Calculate \( s = k^{-1} (z + rd_A) \mod n \). If \( s = 0 \) go back to step 3.
7. The signature is the pair \((r, s)\).

Steps 4 to 7 are accelerated by PKA using:
- \textit{ECDSA sign}
or
- All of the operations below:
  - \textit{ECC Fp scalar multiplication} \( k \times P \)
  - \textit{Modular reduction} \( A \mod n \)
  - \textit{Modular inversion} \( A^{-1} \mod n \)
  - \textit{Modular addition} and \textit{Modular and Montgomery multiplication}

ECDSA signature verification

ECDSA (elliptic curve digital signature algorithm) signature verification function principle is the following: Bob, to authenticate Alice’s signature, must have a copy of her public key curve point \( Q_A \).

Bob can verify that \( Q_A \) is a valid curve point going through the following steps:
1. check that \( Q_A \) is not equal to the identity element \( O \)
2. check that \( Q_A \) is on the agreed curve
3. check that \( n \times Q_A = O \).

Then Bob follows the procedure detailed below:
1. verify that $r$ and $s$ are integer in $[1, n-1]$
2. calculate $e = \text{HASH}(m)$, where $\text{HASH}$ is the agreed cryptographic hash function
3. let $z$ be the $L_n$ leftmost bits of $e$
4. calculate $w = s^{-1} \mod n$
5. calculate $u_1 = zw \mod n$ and $u_2 = rw \mod n$
6. calculate the curve point $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$
7. the signature is valid if $r = x_1 \mod n$, it is invalid otherwise.

Steps 4 to 7 are accelerated by PKA using ECDSA verification.

36.3.6 PKA procedure to perform an operation

Enabling/disabling PKA

Setting the EN bit to 1 in PKA_CR register enables the PKA peripheral. The PKA becomes available when INITOK bit is set in PKA_SR. When EN = 0, the PKA peripheral is kept under reset, with PKA memory still accessible by the application through the AHB interface.

Note: When PKA is in the process of clearing its memory EN bit cannot be set.

Note: When setting EN bit in PKA_CR make sure that the value of MODE bitfield corresponds to an authorized PKA operation (see OPERRF in Section 36.3.7).

Clearing EN bit to 0 while a calculation is in progress causes the operation to be aborted. In this case, the content of the PKA memory is not guaranteed, with the exception of the PKA modes 0x03, 0x20 and 0x24. For these operations, the PKA memory is cleared after abort, making the memory unavailable for 667 cycles. During this clearing time only PKA registers can be accessed, with writes to EN bits ignored.

If INITOK bit stays at 0, make sure that the RNG peripheral is clocked and properly initialized, then try to enable PKA again.

Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in Section 36.4.

Executing a PKA operation

Each of the supported PKA operation is executed using the following procedure:

1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE field of PKA_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set to 1, indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

Note: When PKA is busy (BUSY = 1) any access by the application to PKA RAM is ignored, and the flag RAMERRF is set in PKA_SR.

Selecting an illegal or unknown operation in step 2 triggers an OPERRF error, and step 3 (PROCENDF = 1) never happens. See Section 36.3.7 for details.
Using precomputed Montgomery parameters (PKA Fast mode)

As explained in Section 36.3.4, when computing many operations with the same modulus it can be beneficial for the application to compute only once the corresponding Montgomery parameter (see, for example, Section 36.4.5). This is known as “Fast mode”.

To manage the usage of Fast mode it is recommended to follow the procedure described below:

1. Load in PKA RAM the modulus size and value information. Such information is compiled in Section 36.5.1.
2. Program in PKA_CR register the PKA in Montgomery parameter computation mode (MODE=“0x1”) then assert the START bit.
3. Wait until the PROCENDF bit in the PKA_SR register is set to 1, then read back from PKA memory the corresponding Montgomery parameter, and then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.
4. Proceed with the required PKA operation, loading on top of regular input data the Montgomery information R2 mod m. All addresses are indicated in Section 36.4.

36.3.7 PKA error management

When PKA is used some errors can occur:

- The access to PKA RAM falls outside the expected range. In this case the Address Error flag (ADDRERRF) is set in the PKA_SR register.
- An AHB access to the PKA RAM occurred while the PKA core was using it. In this case the RAM Error Flag (RAMERRF) is set in the PKA_SR register, reads to PKA RAM return zero, while writes are ignored.
- The selected operating mode using MODE bitfield is not listed in PKA operating modes (or in bitfield description), or PKA is running in limited mode (see LMF bit in PKA_SR). In this case the operation error flag (OPERRF) is set in the PKA_SR register, and write to MODE bitfield is ignored.

For each error flag above PKA generates an interrupt if the application sets the corresponding bit in PKA_CR register (see Section 36.6 for details).

ADDRERRF, OPERRF and RAMERRF errors are cleared by setting the corresponding bit in PKA_CLRFR.

The PKA can be re-initialized at any moment by resetting the EN bit in the PKA_CR register.

OPERRF error must be cleared using OPERRFC bit in PKA_CLRFR before a new operation is written in PKA_CR register.

36.4 PKA operating modes

36.4.1 Introduction

The various operations supported by PKA are described in the following subsections, defining the format of the input data and of the results, both stored in the PKA RAM.
Warning: The validity of all input parameters to the PKA must be checked before starting any operation, as PKA assumes that all of them are valid and consistent with each other. Input parameters must not exceed the operand size specified in the operation tables.

The following information applies to all PKA operations.

- PKA core processes 64-bit words in its RAM. Hence hereafter all word size is 64-bit.
- When an element is written as input in the PKA RAM, an additional word with all bits equal to zero has to be added after the most significant input word. This rule does not apply if the operand has a fixed size of 1.
- All reported RAM storage addresses refer to the least significant word of the data, and to obtain the actual address to use application must add to the indicated offset the base address of the PKA.
- Supported operand "Size" are:
  - ROS (RSA operand Size): data size is \( \frac{\text{rsa\_size}}{64} + 1 \) words, with \( \text{rsa\_size} \) equal to the chosen modulus length in bits. For example, when computing RSA with an operand size of 1024 bits, ROS is equal to 17 words, or 1088 bits.
  - EOS (ECC operand Size): data size is \( \frac{\text{ecc\_size}}{64} + 1 \) words, with \( \text{ecc\_size} \) equal to the chosen prime modulus length in bits. For example, when computing ECC with an operand size of 192 bits, EOS is equal to 4 words, or 256 bits.
  - ROS and EOS values include the required additional all 0 word.
- Unless indicated otherwise, all operands in the tables are integers.

Note: Fractional results for above formulas must be rounded up to the nearest integer since PKA core processes 64-bit words.

Note: The maximum ROS is 66 words (4160-bit max exponent size), while the maximum EOS is 11 words (640-bit max operand size).

As a first example (and to better understand the endianess in PKA memory), to prepare the operation \textbf{ECC Fp scalar multiplication}, when the application writes the \( x \) coordinate of point \( P \) for an ECC P256 curve (EOS = 5 words), the least significant bit must be placed in bit 0 at address offset 0x578, and the most significant bit in bit 63 at address offset 0x590. Then, as mentioned above, the application must write the empty word 0x0000000000000000 at address offset 0x598.

As a second example, still to prepare the operation ECC Fp scalar multiplication, when the application need to write the information \( a = -3 \), on a curve with a modulus length of 224 bits (i.e. four 64-bit words, rounded up, plus one) following data must be written in PKA memory:

\[
\begin{align*}
\text{\@RAM}+410 &\quad 0x0000000000000001 & \quad /* \text{curve coefficient 'a' sign without extra word} */ \\
\text{\@RAM}+418 &\quad 0x0000000000000011 & \quad /* \text{value of |a| LSB} */ \\
\text{\@RAM}+420 &\quad 0x0000000000000000 & \quad ... \\
\text{\@RAM}+428 &\quad 0x0000000000000000 & \quad ...
\end{align*}
\]
36.4.2 Montgomery parameter computation

This function is used to compute the Montgomery parameter \( R^2 \mod n \) used by PKA to convert operands into the Montgomery residue system representation. This operation can be very useful when fast mode operation is used, because in this case the Montgomery parameter is passed as input, saving the time for its computation.

Note: This operation can also be used with ECC curves. In this case prime modulus length and EOS size must be used.

Operation instructions for Montgomery parameter computation are summarized in Table 336.

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MODE 0x01</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Modulus length</td>
<td>(in bits, ( 0 \leq \text{value} &lt; 4160 ))</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>Modulus value n</td>
<td>(odd integer only, ( n &lt; 2^{4160} ))</td>
<td>RAM@0x1088</td>
<td>ROS</td>
</tr>
<tr>
<td>OUT</td>
<td>Result: ( R^2 \mod n )</td>
<td>-</td>
<td>RAM@0x620</td>
</tr>
</tbody>
</table>
36.4.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications the PKA accelerates multiplication which has at least one input in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication \( A \times B \mod n \)

The method to perform above operations are described below. Note that “x” function is this operation, and \( A, B, C \) operands are in the natural domain.

1. Inward (or outward) conversion into (or from) Montgomery domain
   a) Assuming that \( A \) is an integer in the natural domain:
      - Compute \( r^2 \mod n \) using Montgomery parameter computation.
      - Result \( AR = A \times r^2 \mod n \) is \( A \) in the Montgomery domain.
   b) Assuming that \( BR \) is an integer in the Montgomery domain:
      - Result \( B = BR \times 1 \mod n \) is \( B \) in the natural domain.
      - Similarly, above value \( AR \) computed in a) can be converted into the natural domain by computing \( A = AR \times 1 \mod n \).

2. Simple modular multiplication \( A \times B \mod n \)
   a) Compute \( r^2 \mod n \) using Montgomery parameter computation.
   b) Compute \( AR = A \times r^2 \mod n \). Output is in the Montgomery domain.
   c) Compute \( AB = AR \times B \mod n \). Output is in natural domain.

3. Multiple modular multiplication \( A \times B \times C \mod n \)
   a) Compute \( r^2 \mod n \) using Montgomery parameter computation.
   b) Compute \( AR = A \times r^2 \mod n \). Output is in the Montgomery domain.
   c) Compute \( BR = B \times r^2 \mod n \). Output is in the Montgomery domain.
   d) Compute \( ABR = AR \times BR \mod n \). Output is in the Montgomery domain.
   e) Compute \( CR = C \times r^2 \mod n \). Output is in the Montgomery domain.
   f) Compute \( ABCR = ABR \times CR \mod n \). Output is in the Montgomery domain.
   g) (optional) Repeat the two steps above if more operands need to be multiplied.
   h) Compute \( ABC = ABCR \times 1 \mod n \) to retrieve the result in natural domain.

Operation instructions for Montgomery multiplication are summarized in Table 339.

### Table 338. Modular subtraction

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MODE 0x0F</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>IN</td>
<td>Operand length (in bits, not null)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>IN</td>
<td>Operand A ((0 \leq A &lt; n))</td>
<td>RAM@0xA50</td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>Operand B ((0 \leq B &lt; n))</td>
<td>RAM@0xC68</td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>Modulus value n ((n &lt; 2^{4160}))</td>
<td>RAM@0x1088</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result: A-B \mod n ((0 \leq \text{result} &lt; n))</td>
<td>RAM@0xE78</td>
<td></td>
</tr>
</tbody>
</table>
**36.4.6 Modular exponentiation**

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It consists in the computation of \( a^b \mod n \).

RSA operation involving public information (RSA encryption) can use the normal or fast mode detailed on Table 340 and Table 341. RSA operation involving secret information (RSA decryption) must use the protected mode detailed on Table 342, for security reason.

*Note:* Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware.

*Note:* When \( \text{MODE} = 0x03 \), if the error output is different from 0xD60D all the memory content is cleared by PKA to avoid leaking information about the private key.

Operation instructions for modular exponentiation are summarized in Table 340 (normal mode), Table 341 (fast mode) and in Table 342 (protected mode). Fast mode usage is explained in Section 36.3.6.

---

### Table 339. Montgomery multiplication

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MODE</td>
<td>0x10</td>
<td>PKA_CR</td>
</tr>
<tr>
<td>IN</td>
<td>Operand length</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
</tr>
<tr>
<td>IN</td>
<td>Operand A</td>
<td>((0 \leq A &lt; n))</td>
<td>RAM@0xA50</td>
</tr>
<tr>
<td>IN</td>
<td>Operand B</td>
<td>((0 \leq B &lt; n))</td>
<td>RAM@0xC68</td>
</tr>
<tr>
<td>IN</td>
<td>Modulus value n</td>
<td>(odd integer only, ( n &lt; 2^{4160} ))</td>
<td>RAM@0x1088</td>
</tr>
<tr>
<td>OUT</td>
<td>Result: ( A \times B \mod n )</td>
<td>-</td>
<td>RAM@0xE78</td>
</tr>
</tbody>
</table>

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples 2 and 3).

### Table 340. Modular exponentiation (normal mode)

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MODE</td>
<td>0x00</td>
<td>PKA_CR</td>
</tr>
<tr>
<td>IN</td>
<td>Exponent length</td>
<td>(in bits, not null)</td>
<td>RAM@0x400</td>
</tr>
<tr>
<td>IN</td>
<td>Operand length</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
</tr>
<tr>
<td>IN/OUT</td>
<td>Operand A (base of exponentiation)</td>
<td>((0 \leq A &lt; n))</td>
<td>RAM@0xC68</td>
</tr>
<tr>
<td>IN</td>
<td>Exponent e</td>
<td>((0 \leq e &lt; n))</td>
<td>RAM@0xE78</td>
</tr>
<tr>
<td>IN</td>
<td>Modulus value n</td>
<td>(odd integer only, ( n &lt; 2^{4160} ))</td>
<td>RAM@0x1088</td>
</tr>
<tr>
<td>OUT</td>
<td>Result: ( A^e \mod n )</td>
<td>((0 \leq \text{result} &lt; n))</td>
<td>RAM@0x838</td>
</tr>
</tbody>
</table>
Modular inversion

Modular inversion operation consists in the computation of multiplicative inverse $A^{-1} \mod n$. If the modulus $n$ is prime, for all values of $A$ (1 ≤ $A < n$) modular inversion output is valid. If the modulus $n$ is not prime, $A$ has an inverse only if the greatest common divisor between $A$ and $n$ is 1.

If the operand $A$ is a divisor of the modulus $n$ the result is a multiple of a factor of $n$.

Operation instructions for modular inversion are summarized in Table 343.

---

Table 341. Modular exponentiation (fast mode)

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MODE 0x02</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>IN</td>
<td>Exponent length</td>
<td>(in bits, not null)</td>
<td>RAM@0x400</td>
</tr>
<tr>
<td>IN</td>
<td>Operand length</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
</tr>
<tr>
<td>IN/OIUT</td>
<td>Operand A (base of exponentiation)</td>
<td>(0 ≤ $A &lt; n$)</td>
<td>RAM@0xC68</td>
</tr>
<tr>
<td>IN</td>
<td>Exponent $e$</td>
<td>(0 ≤ $e &lt; n$)</td>
<td>RAM@0xE78</td>
</tr>
<tr>
<td>IN</td>
<td>Modulus value $n$</td>
<td>(odd integer only, $n &lt; 2^{4160}$)</td>
<td>RAM@0x1088</td>
</tr>
<tr>
<td>IN/OIUT</td>
<td>Montgomery parameter $R2 \mod n$</td>
<td>(mandatory)</td>
<td>RAM@0x620</td>
</tr>
<tr>
<td>OUT</td>
<td>Result: $A^e \mod n$</td>
<td>(0 ≤ result &lt; $n$)</td>
<td>RAM@0x838</td>
</tr>
</tbody>
</table>

Table 342. Modular exponentiation (protected mode)

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MODE 0x03</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>IN</td>
<td>Exponent $e$ length</td>
<td>(in bits, not null)</td>
<td>RAM@0x400</td>
</tr>
<tr>
<td>IN</td>
<td>Modulus or operand length</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
</tr>
<tr>
<td>IN</td>
<td>Operand A (base of exponentiation)</td>
<td>(0 ≤ $A &lt; n$)</td>
<td>RAM@0x16C8</td>
</tr>
<tr>
<td>IN</td>
<td>Exponent $e$</td>
<td>(0 ≤ $e &lt; n$)</td>
<td>RAM@0x14B8</td>
</tr>
<tr>
<td>IN</td>
<td>Modulus value $n$</td>
<td>(odd integer only, $n &lt; 2^{4096}$)</td>
<td>RAM@0x0838</td>
</tr>
<tr>
<td>IN</td>
<td>Phi value(1)</td>
<td>-</td>
<td>RAM@0x0C68</td>
</tr>
<tr>
<td>OUT</td>
<td>Result: $A^e \mod n$</td>
<td>(0 ≤ result &lt; $n$)</td>
<td>RAM@0x838</td>
</tr>
<tr>
<td>ERROR</td>
<td>Error $A^e \mod n$</td>
<td>- No errors: 0xD60D Errors: 0xCB9C9</td>
<td>RAM@0x1298</td>
</tr>
</tbody>
</table>

---

(1) Euler totient function of $n^1$ with $phi = (p - 1) \times (q - 1)$, where $p$ and $q$ are prime factors of modulus $n$ (see NIST SP800-56B Appendix C or RSA encryption/decryption principle for details). As optimization it is recommended to keep phi information as part of the key pair generation. Alternative is to store the key as $(p, q, e, d)$ instead of $(N, e, d, phi)$; in this case, to derive $N$ and phi using PKA arithmetic multiplier:

$$N = p \times q,$$

and

$$phi = (p - 1) \times (q - 1).$$
36.4.8 Modular reduction

Modular reduction operation consists in the computation of the remainder of $A$ divided by $n$. Operation instructions are summarized in Table 344.

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x0D</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length</td>
<td>(in bits, not null)</td>
<td>RAM@0x400</td>
<td>64 bits</td>
</tr>
<tr>
<td>Modulus length</td>
<td>(in bits, 8 &lt; value &lt; 4160)</td>
<td>RAM@0x408</td>
<td></td>
</tr>
<tr>
<td>Operand A</td>
<td>$(0 \leq A &lt; n)$</td>
<td>RAM@0xA50</td>
<td>ROS</td>
</tr>
<tr>
<td>Modulus value $n$</td>
<td>(odd integer only, $n &lt; 2^{4160}$)</td>
<td>RAM@0xC68</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result: $A^{-1} \mod n$</td>
<td>0 &lt; result &lt; $n$</td>
<td>RAM@0xE78</td>
</tr>
</tbody>
</table>

36.4.9 Arithmetic addition

Arithmetic addition operation consists in the computation of $A + B$. Operation instructions are summarized in Table 345.

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x09</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length M</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>Operand A</td>
<td>$(0 \leq A &lt; 2^M)$</td>
<td>RAM@0xA50</td>
<td>ROS</td>
</tr>
<tr>
<td>Operand B</td>
<td>$(0 \leq B &lt; 2^M)$</td>
<td>RAM@0xC68</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result: $A+B$</td>
<td>$(0 \leq \text{result} &lt; 2^{M+1})$</td>
<td>RAM@0xE78</td>
</tr>
</tbody>
</table>

36.4.10 Arithmetic subtraction

Arithmetic subtraction operation consists in the following computations:

- If $A \geq B$ result equals $A - B$
- If $A < B$ and $M/32$ residue is $> 0$ result equals $A + 2^{\text{int}(M/32)^*32+1} - B$
- If $A < B$ and $M/32$ residue is $0$ result equals $A + 2^{\text{int}(M/32)^*32} - B$
For the last two bullets the 32-bit word following the most significant word of the output equals 0xFFFF FFFF, as result is negative.

Operation instructions are summarized in Table 346.

**36.4.11 Arithmetic multiplication**

Arithmetic multiplication operation consists in the computation of AxB. Operation instructions are summarized in Table 347.

**Table 346. Arithmetic subtraction**

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MODE</td>
<td>0x0A</td>
<td>PKA_CR 6 bits</td>
</tr>
<tr>
<td></td>
<td>Operand length M</td>
<td>(in bits, not null)</td>
<td>RAM@0x408 64 bits</td>
</tr>
<tr>
<td></td>
<td>Operand A</td>
<td>(0 ≤ A &lt; 2^M)</td>
<td>RAM@0xA50</td>
</tr>
<tr>
<td></td>
<td>Operand B</td>
<td>(0 ≤ B &lt; 2^M)</td>
<td>RAM@0x68</td>
</tr>
<tr>
<td>OUT</td>
<td>Result: A-B</td>
<td>(0 ≤ result &lt; 2^M)</td>
<td>RAM@0xE78</td>
</tr>
</tbody>
</table>

**Table 347. Arithmetic multiplication**

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MODE</td>
<td>0x0B</td>
<td>PKA_CR 6 bits</td>
</tr>
<tr>
<td></td>
<td>Operand length M</td>
<td>(in bits, not null)</td>
<td>RAM@0x408 64 bits</td>
</tr>
<tr>
<td></td>
<td>Operand A</td>
<td>(0 ≤ A &lt; 2^M)</td>
<td>RAM@0xA50</td>
</tr>
<tr>
<td></td>
<td>Operand B</td>
<td>(0 ≤ B &lt; 2^M)</td>
<td>RAM@0x68</td>
</tr>
<tr>
<td>OUT</td>
<td>Result: AxB</td>
<td>(0 ≤ result &lt; 2^M)</td>
<td>RAM@0xE78 2xROS</td>
</tr>
</tbody>
</table>

**36.4.12 Arithmetic comparison**

Arithmetic comparison operation consists in the following computation:
- If A = B then result = 0xED2C
- If A > B then result = 0x7AF8
- If A < B then result = 0x916A

Operation instructions for arithmetic comparison are summarized in Table 348.

**Table 348. Arithmetic comparison**

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MODE</td>
<td>0x0C</td>
<td>PKA_CR 6 bits</td>
</tr>
<tr>
<td></td>
<td>Operand length M</td>
<td>(in bits, not null)</td>
<td>RAM@0x408 64 bits</td>
</tr>
<tr>
<td></td>
<td>Operand A</td>
<td>(0 ≤ A &lt; 2^M)</td>
<td>RAM@0xA50</td>
</tr>
<tr>
<td></td>
<td>Operand B</td>
<td>(0 ≤ B &lt; 2^M)</td>
<td>RAM@0x68</td>
</tr>
<tr>
<td>OUT</td>
<td>Result A?B</td>
<td>0xED2C, 0x7AF8 or 0x916A</td>
<td>RAM@0xE78 64 bits</td>
</tr>
</tbody>
</table>
36.4.13 RSA CRT exponentiation

For efficiency many popular crypto libraries such as OpenSSL RSA use the following optimization for decryption and signing based on the chinese remainder theorem (CRT):

- $p$ and $q$ are precomputed primes, stored as part of the private key
- $d_P = d \mod (p - 1)$
- $d_Q = d \mod (q - 1)$ and
- $q_{inv} = q^{-1} \mod p$

These values allow the recipient to compute the exponentiation $m = A^d \mod pq$ more efficiently as follows:

- $m_1 = A^{d_P} \mod p$
- $m_2 = A^{d_Q} \mod q$
- $h = q_{inv} (m_1 - m_2) \mod p$, with $m_1 > m_2$
- $m = m_2 + hq \mod pq$

Operation instructions for computing CRT exponentiation $A^d \mod pq$ are summarized in Table 349.

### Table 349. CRT exponentiation

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN MODE</td>
<td>0x07</td>
<td>PKA.CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>IN Operand length</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>IN Operand $d_p$</td>
<td>$(0 &lt; d_p &lt; 2^{M/2})$</td>
<td>RAM@0x730</td>
<td></td>
</tr>
<tr>
<td>IN Operand $d_Q$</td>
<td>$(0 &lt; d_Q &lt; 2^{M/2})$</td>
<td>RAM@0xE78</td>
<td></td>
</tr>
<tr>
<td>IN Operand $q_{inv}$</td>
<td>$(0 &lt; q_{inv} &lt; 2^{M/2})$</td>
<td>RAM@0x948</td>
<td></td>
</tr>
<tr>
<td>IN Prime $p^{(1)}$</td>
<td>$(0 &lt; p &lt; 2^{M/2})$</td>
<td>RAM@0xB60</td>
<td></td>
</tr>
<tr>
<td>IN Prime $q^{(1)}$</td>
<td>$(0 &lt; q &lt; 2^{M/2})$</td>
<td>RAM@0x1088</td>
<td></td>
</tr>
<tr>
<td>IN Operand $A$</td>
<td>$(0 &lt; A &lt; 2^M)$</td>
<td>RAM@0x12A0</td>
<td>ROS</td>
</tr>
<tr>
<td>OUT Result: $A^d \mod pq$</td>
<td>$(0 \leq \text{result} &lt; pq)$</td>
<td>RAM@0x838</td>
<td>ROS</td>
</tr>
</tbody>
</table>

1. Must be different from 2.

36.4.14 Point on elliptic curve Fp check

This operation consists in checking whether a given point $P(x, y)$ satisfies or not the curves over prime fields equation $y^2 = (x^3 + ax + b) \mod p$, where $a$ and $b$ are elements of the curve.

Operation instructions for point on elliptic curve Fp check are summarized in Table 350.
36.4.15  ECC Fp scalar multiplication

This operation consists in the computation of a k x P (x_P, y_P), where P is a point on a curve over prime fields and “x” is the elliptic curve scalar point multiplication. Result of the computation is a point that belongs to the same curve or a point at infinity.

Operation instructions for ECC Fp scalar multiplication are summarized in Table 351.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events, a PKA tamper event is triggered to TAMPER peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation "Point on elliptic curve" can be used to avoid this.

Table 350. Point on elliptic curve Fp check

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x28</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Modulus length</td>
<td>(in bits, not null, 8 &lt; value &lt; 640)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>Curve coefficient a sign</td>
<td>0x0: positive 0x1: negative</td>
<td>RAM@0x410</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient [a]</td>
<td>(absolute value,</td>
<td>RAM@0x418</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient b</td>
<td>([b] &lt; p)</td>
<td>RAM@0x520</td>
<td></td>
</tr>
<tr>
<td>Curve modulus value p</td>
<td>(odd integer prime, 0 &lt; p &lt; 2640)</td>
<td>RAM@0x470</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate x</td>
<td>(x &lt; p)</td>
<td>RAM@0x578</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate y</td>
<td>(y &lt; p)</td>
<td>RAM@0x5D0</td>
<td></td>
</tr>
<tr>
<td>Montgomery parameter R2 mod n</td>
<td>-</td>
<td>RAM@0x4C8</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Result: point P on curve</td>
<td>– 0xD60D: point on curve – 0xA3B7: point not on curve – 0xF946: x or y coordinate is not smaller than modulus p</td>
<td>RAM@0x680</td>
<td>64 bits</td>
</tr>
</tbody>
</table>

Table 351. ECC Fp scalar multiplication

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x20</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Curve prime order n length</td>
<td>(in bits, not null,)</td>
<td>RAM@0x400</td>
<td></td>
</tr>
<tr>
<td>Curve modulus p length</td>
<td>(in bits, not null, 8 &lt; value &lt; 640)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>Curve coefficient a sign</td>
<td>– 0x0: positive – 0x1: negative</td>
<td>RAM@0x410</td>
<td></td>
</tr>
</tbody>
</table>
When performing this operation the following special cases must be noted:

- For \( k = 0 \) this function returns a point at infinity \((0, 0)\) if curve parameter \( b \) is nonzero, \((0, 1)\) otherwise. For \( k \) different from 0 it might happen that a point at infinity is returned. When the application detects this behavior a new computation must be carried out.

- For \( k < 0 \) (i.e. a negative scalar multiplication is required) the multiplier absolute value \( k = |-k| \) must be provided to the PKA. After the computation completion, the formula \(-P = (x, -y)\) can be used to compute the \( y \) coordinate of the effective final result (the \( x \) coordinate remains the same).

Note: If the error output is different from 0xD60D all the memory content is cleared by PKA to avoid leaking information about the private key.

### 36.4.16 ECDSA sign

ECDSA signing operation (outlined in Section 36.3.5) is summarized in Table 352 (input parameters) and in Table 353 (output parameters).

The application has to check if the output error is equal to 0xD60D, if it is different a new \( k \) must be generated and the ECDSA sign operation must be repeated.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation "Point on elliptic curve" can be used to avoid this.
Public key accelerator (PKA) RM0481

1. **Note:** If the error output equals 0xD60D or 0xCBC9 all the memory content is cleared by PKA to avoid leaking information about the private key. If error output equals 0xA3B7 or 0xF946 PKA memory content is partially erased, keeping the error code readable.

### Extended ECDSA support

PKA also supports extended ECDSA signature, for which the inputs and the outputs are the same as ECDSA signature (Table 352 and Table 353, respectively), with the addition of the coordinates of the point kG. This extra output is defined in Table 354.

---

### Table 352. ECDSA sign - Inputs

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE</td>
<td>0x24</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Curve prime order n length ( (nlen) )</td>
<td>(in bits, not null)</td>
<td>RAM@0x400</td>
<td>64 bits</td>
</tr>
<tr>
<td>Curve modulus p length</td>
<td>(in bits, ( 8 &lt; \text{value} &lt; 640 ))</td>
<td>RAM@0x408</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient a sign</td>
<td>0x0: positive 0x1: negative</td>
<td>RAM@0x410</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient ([a])</td>
<td>(absolute value, (</td>
<td>a</td>
<td>&lt; p ))</td>
</tr>
<tr>
<td>Curve coefficient b</td>
<td>(positive integer)</td>
<td>RAM@0x520</td>
<td></td>
</tr>
<tr>
<td>Curve modulus value ( p )</td>
<td>(odd integer prime, ( 0 &lt; p &lt; 2^{640} ))</td>
<td>RAM@0x1088</td>
<td></td>
</tr>
<tr>
<td>Integer ( k^{(1)} )</td>
<td>( 0 \leq k &lt; 2^{640} )</td>
<td>RAM@0x12A0</td>
<td></td>
</tr>
<tr>
<td>Curve base point G coordinate ( x )</td>
<td>( x &lt; p )</td>
<td>RAM@0x578</td>
<td></td>
</tr>
<tr>
<td>Curve base point G coordinate ( y )</td>
<td>( y &lt; p )</td>
<td>RAM@0x470</td>
<td></td>
</tr>
<tr>
<td>Hash of message ( z )</td>
<td>(hash size equal to ( nlen ))(^{(2)} )</td>
<td>RAM@0xFE8</td>
<td></td>
</tr>
<tr>
<td>Private key ( d )</td>
<td>( 0 &lt; d )</td>
<td>RAM@0xF28</td>
<td></td>
</tr>
<tr>
<td>Curve prime order n</td>
<td>(integer prime)</td>
<td>RAM@0xF88</td>
<td></td>
</tr>
</tbody>
</table>

\(^{(1)}\) This integer is usually a cryptographically secure random number, but in some cases \( k \) can be deterministically generated.

\(^{(2)}\) Padding with zeroes or hash truncation must be used to have the hash parameter size equal to the curve prime order n length.

### Table 353. ECDSA sign - Outputs

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT</td>
<td></td>
<td></td>
<td>EOS</td>
</tr>
<tr>
<td>Signature part ( r )</td>
<td>( 0 &lt; r &lt; n )</td>
<td>RAM@0x730</td>
<td></td>
</tr>
<tr>
<td>Signature part ( s )</td>
<td>( 0 &lt; s &lt; n )</td>
<td>RAM@0x788</td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td>Result of signature</td>
<td></td>
<td>64 bits</td>
</tr>
<tr>
<td></td>
<td>(- 0xD60D: \text{successful computation, no error})</td>
<td>RAM@0xFE0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(- 0xCBC9: \text{failed computation})</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(- 0xA3B7: \text{signature part } r \text{ is equal to } 0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(- 0xF946: \text{signature part } s \text{ is equal to } 0)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If the error output equals 0xD60D or 0xCBC9 all the memory content is cleared by PKA to avoid leaking information about the private key. If error output equals 0xA3B7 or 0xF946 PKA memory content is partially erased, keeping the error code readable.
ECDSA verification

ECDSA verification operation (outlined in Section 36.3.5) is summarized in Table 355 (input parameters) and Table 356 (output parameters).

The application has to check if the output error is equal to 0xD60D, if different the signature is not verified.

### Table 354. Extended ECDSA sign - Extra outputs

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Curve point kG coordinate x₁</td>
<td>(0 ≤ x₁ &lt; p)</td>
<td>RAM@0x1400</td>
<td></td>
</tr>
<tr>
<td>Curve point kG coordinate y₁</td>
<td>(0 ≤ y₁ &lt; p)</td>
<td>RAM@0x1458</td>
<td></td>
</tr>
</tbody>
</table>

### Table 355. ECDSA verification - Inputs

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x26</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Curve prime order n length (nlen)</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>Curve modulus p length</td>
<td>(in bits, not null, 8 &lt; value &lt; 640)</td>
<td>RAM@0x4C8</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient a sign</td>
<td>0x0: positive</td>
<td>RAM@0x468</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1: negative</td>
<td>RAM@0x468</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient</td>
<td>(absolute value,</td>
<td>RAM@0x470</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Curve modulus value p</td>
<td>(odd integer prime, 0 &lt; p &lt; 2⁶⁴⁰)</td>
<td>RAM@0x4D0</td>
<td></td>
</tr>
<tr>
<td>Curve base point G coordinate x</td>
<td>(x &lt; p)</td>
<td>RAM@0x678</td>
<td></td>
</tr>
<tr>
<td>Curve base point G coordinate y</td>
<td>(y &lt; p)</td>
<td>RAM@0x6D0</td>
<td>EOS</td>
</tr>
<tr>
<td>Public-key curve point Q coordinate x_Q</td>
<td>(x_Q &lt; p)</td>
<td>RAM@0x12F8</td>
<td></td>
</tr>
<tr>
<td>Public-key curve point Q coordinate y_Q</td>
<td>(y_Q &lt; p)</td>
<td>RAM@0x1350</td>
<td></td>
</tr>
<tr>
<td>Signature part r</td>
<td>(0 &lt; r &lt; n)</td>
<td>RAM@0x10E0</td>
<td></td>
</tr>
<tr>
<td>Signature part s</td>
<td>(0 &lt; s &lt; n)</td>
<td>RAM@0xC68</td>
<td></td>
</tr>
<tr>
<td>Hash of message z</td>
<td>(hash size equal to nlen)¹</td>
<td>RAM@0x13A8</td>
<td></td>
</tr>
<tr>
<td>Curve prime order n</td>
<td>(integer prime)</td>
<td>RAM@0x1088</td>
<td></td>
</tr>
</tbody>
</table>

¹. Padding with zeroes or hash truncation must be used to have the hash parameter size equal to the curve prime order n length.
36.4.18 **ECC complete addition**

ECC complete addition computes the addition of two given points on an elliptic curve.

Operation instructions are summarized in **Table 357**.

*Note:* The two input points and the resulting point are represented in Jacobian coordinates \((X,Y,Z)\). To input a point in affine coordinates \((x,y)\) conversion \((X,Y,Z) = (x,y,1)\) can be used. To convert resulting point to Jacobian coordinates conversion \((x,y) = (X/Z^2, Y/Z^3)\) can be used.

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT</td>
<td>Result: ECDSA verify</td>
<td>RAM@0x5D0</td>
<td>64 bits</td>
</tr>
<tr>
<td></td>
<td>– 0xD60D: valid signature</td>
<td>RAM@0x578</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>– 0xA3B7: invalid signature</td>
<td>RAM@0x578</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>Computed signature part (r)</td>
<td>RAM@0x578</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>((0 &lt; r &lt; n))</td>
<td>RAM@0x578</td>
<td>EOS</td>
</tr>
</tbody>
</table>

**Table 357. ECC complete addition**

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td><strong>MODE</strong></td>
<td>0x23</td>
<td>6 bits</td>
</tr>
<tr>
<td></td>
<td>Curve modulus (p) length</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td></td>
<td>((\text{in bits, not null, }8&lt;\text{value}&lt;640))</td>
<td>RAM@0x410</td>
<td>64 bits</td>
</tr>
<tr>
<td></td>
<td>Curve coefficient (a) sign</td>
<td>RAM@0x418</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>– 0x0: positive</td>
<td>RAM@0x418</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>– 0x1: negative</td>
<td>RAM@0x418</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>Curve modulus value (p)</td>
<td>RAM@0x470</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>((\text{odd integer prime, }0&lt;p&lt;2^{640}))</td>
<td>RAM@0x470</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>Curve coefficient (a)</td>
<td>RAM@0x470</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>((\text{absolute value, }</td>
<td>a</td>
<td>&lt;p))</td>
</tr>
<tr>
<td></td>
<td>First point P coordinate (X)</td>
<td>RAM@0x628</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>((x&lt;p))</td>
<td>RAM@0x628</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>First point P coordinate (Y)</td>
<td>RAM@0x680</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>((y&lt;p))</td>
<td>RAM@0x680</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>First point P coordinate (Z)</td>
<td>RAM@0x6D8</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>((z&lt;p))</td>
<td>RAM@0x6D8</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>Second point Q coordinate (X)</td>
<td>RAM@0x730</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>((x&lt;p))</td>
<td>RAM@0x730</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>Second point Q coordinate (Y)</td>
<td>RAM@0x788</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>((y&lt;p))</td>
<td>RAM@0x788</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>Second point Q coordinate (Z)</td>
<td>RAM@0x7E0</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>((z&lt;p))</td>
<td>RAM@0x7E0</td>
<td>EOS</td>
</tr>
<tr>
<td>OUT</td>
<td>Result coordinate (X)</td>
<td>RAM@0x6D60</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>((x&lt;p))</td>
<td>RAM@0x6D60</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>Result coordinate (Y)</td>
<td>RAM@0xDB8</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>((y&lt;p))</td>
<td>RAM@0xDB8</td>
<td>EOS</td>
</tr>
<tr>
<td></td>
<td>Result coordinate (Z)</td>
<td>RAM@0xE10</td>
<td>EOS</td>
</tr>
</tbody>
</table>
Note: The two input points are represented in Jacobian coordinates \((X, Y, Z)\). To input a point in affine coordinates \((x, y)\) conversion \((X, Y, Z) = (x, y, 1)\) can be used. The result is represented in affine coordinates \((x, y)\).

### ECC projective to affine

ECC projective to affine operation computes the conversion between the representation of a point \(P\) in homogeneous projective coordinates and the representation of the point \(P\) in affine coordinates. Namely, if the point is represented by the triple \((X, Y, Z)\), it computes the affine coordinates \((x, y) = (X/Z, Y/Z)\).

All the operations are performed modulo the modulus \(p\) of the curve, which the point belongs to. If the resulting point is the point at infinity (error code 0xA3B7), the resulting coordinate equals \((0, 0)\).

Operation instructions are summarized in Table 359.
36.5 Example of configurations and processing times

36.5.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curves can be described with a short Weierstrass equation, \( y^2 = x^3 + ax + b \ (\text{mod} \ p) \).

**Note:** Binary curves, Edwards curves and Curve25519 are not supported by the PKA. The maximum supported operand size for ECC operations is 640 bits.

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer \( p \), used as the modulus for all point arithmetic in the finite field \( \text{GF}(p) \)
- the (usually prime) integer \( n \), the order of the group generated by \( G \), defined below
- the base point of the curve \( G \), defined by its coordinates \((G_x, G_y)\)
- the integers \( a \) and \( b \), coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define \( a \) as negative, PKA supports two representations:

1. **a defined as \( p-a \)** in the finite field \( \text{GF}(p) \), for example \( p-3 \):
   - Curve coefficient \( p = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF \)
   - Curve coefficient \( a \) sign= 0x0 (positive)
   - Curve coefficient \( a = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF \)

2. **a defined as negative**, for example \(-3\):
   - Curve coefficient \( p = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF \)
   - Curve coefficient \( a \) sign= 0x1 (negative)
   - Curve coefficient \( a = 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 \)

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x2F</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Curve modulus ( p ) length</td>
<td>(in bits, 8 &lt; value &lt; 640)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>Curve modulus value ( p )</td>
<td>(odd integer prime, 0 &lt; ( p ) &lt; 2^{640})</td>
<td>RAM@0x470</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate X (projective)</td>
<td>((x &lt; p))</td>
<td>RAM@0x0D60</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate Y (projective)</td>
<td>((y &lt; p))</td>
<td>RAM@0xDB8</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate Z (projective)</td>
<td>((z &lt; p))</td>
<td>RAM@0xE10</td>
<td></td>
</tr>
<tr>
<td>Montgomery parameter R2 mod ( n )</td>
<td>-</td>
<td>RAM@0x4C8</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point P coordinate x (affine)</td>
<td>((x &lt; p))</td>
<td>RAM@0x578</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate y (affine)</td>
<td>((y &lt; p))</td>
<td>RAM@0x5D0</td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error code</td>
<td></td>
<td>RAM@0x680</td>
<td>64 bits</td>
</tr>
</tbody>
</table>

Table 359. ECC projective to affine
Table 360 summarizes the family of curves supported by PKA for ECC operations.

<table>
<thead>
<tr>
<th>Curve name</th>
<th>Standard</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-192</td>
<td>NIST</td>
<td>Digital Signature Standard (DSS), NIST FIPS 186-4</td>
</tr>
<tr>
<td>P-224</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P-256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P-384</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P-521</td>
<td></td>
<td></td>
</tr>
<tr>
<td>brainpoolP224r1, brainpoolP224t1</td>
<td>IETF</td>
<td>– Brainpool Elliptic Curves, IETF RFC 5639 – Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry, IETF RFC 6932 <a href="https://tools.ietf.org">https://tools.ietf.org</a></td>
</tr>
<tr>
<td>brainpoolP256r1, brainpoolP256t1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>brainpoolP320r1, brainpoolP320t1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>brainpoolP384r1, brainpoolP384t1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>brainpoolP512r1, brainpoolP512t1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>secp192k1, secp192r1</td>
<td>SEC</td>
<td>Standards for Efficient Cryptography SEC 2 curves <a href="https://www.secg.org">https://www.secg.org</a></td>
</tr>
<tr>
<td>secp224k1, secp224r1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>secp256k1, secp256r1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>secp384r1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>secp521r1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recommended curve parameters for public key cryptographic algorithm SM2</td>
<td>OSCCA</td>
<td>– Public key cryptographic algorithm SM2 based on elliptic curves, Organization of State Commercial Administration of China OSCCA SM2, December 2010 – Digital signatures - Part 3 Discrete logarithm based mechanisms, ISO/IEC 14888-3, November 2018</td>
</tr>
</tbody>
</table>
36.5.2 Computation times

The following tables summarize the PKA computation times, expressed in AHB clock cycles.

### Table 361. Modular exponentiation

<table>
<thead>
<tr>
<th>Exponent length (in bits)</th>
<th>Mode</th>
<th>Modulus length (in bits)</th>
<th>1024</th>
<th>2048</th>
<th>3072</th>
<th>4096</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
<td>124600</td>
<td>491000</td>
<td>684000</td>
<td>1133200</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>22700</td>
<td>82000</td>
<td>178000</td>
<td>311000</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Normal</td>
<td>135700</td>
<td>531400</td>
<td>772400</td>
<td>1288000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>33800</td>
<td>122500</td>
<td>266500</td>
<td>465800</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Normal</td>
<td>180000</td>
<td>693700</td>
<td>1126200</td>
<td>1907200</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>78200</td>
<td>284700</td>
<td>620400</td>
<td>1085000</td>
<td></td>
</tr>
<tr>
<td>$2^{16} + 1$</td>
<td>Normal</td>
<td>9958000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>5850000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CRT(1)</td>
<td>5748000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>Protected</td>
<td>-</td>
<td>9958000</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>-</td>
<td>5850000</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>-</td>
<td>5748000</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CRT(1)</td>
<td>-</td>
<td>1775000</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>2048</td>
<td>Protected</td>
<td>-</td>
<td>6388600</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>-</td>
<td>4224000</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>-</td>
<td>4183200</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CRT(1)</td>
<td>-</td>
<td>1167000</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>3072</td>
<td>Protected</td>
<td>-</td>
<td>-</td>
<td>199403000</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>-</td>
<td>-</td>
<td>136830000</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>-</td>
<td>-</td>
<td>136325000</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CRT(1)</td>
<td>-</td>
<td>-</td>
<td>36886000</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>4096</td>
<td>Protected</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>454318000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Normal</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>316000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>315226000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CRT(1)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>84577000</td>
<td></td>
</tr>
</tbody>
</table>

1. CRT stands for chinese remainder theorem optimization (MODE bitfield= 0x07).

### Table 362. ECC scalar multiplication(1)

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>521</th>
<th>640</th>
</tr>
</thead>
<tbody>
<tr>
<td>160</td>
<td>1590000</td>
<td>3083000</td>
<td>5339000</td>
<td>8518000</td>
<td>17818000</td>
<td>21053000</td>
<td>31826000</td>
<td></td>
</tr>
</tbody>
</table>

1. These times depend on the number of 1s included in the scalar parameter, and include the computation of Montgomery parameter R2.
Table 363. ECDSA signature average computation time<sup>(1) (2)</sup>

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>521</th>
<th>640</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td>1500000</td>
<td>2744000</td>
<td>4579000</td>
<td>7184000</td>
<td>14455000</td>
<td>16685000</td>
<td>24965000</td>
</tr>
</tbody>
</table>

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.
2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

Table 364. ECDSA verification average computation times

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>521</th>
<th>640</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1011000</td>
<td>1495000</td>
<td>2938000</td>
<td>5014000</td>
<td>7979000</td>
<td>16804000</td>
<td>19254000</td>
<td>29582000</td>
</tr>
</tbody>
</table>

Table 365. ECC double base ladder average computation times

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>521</th>
<th>640</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>967000</td>
<td>1419000</td>
<td>2768000</td>
<td>4784000</td>
<td>7547000</td>
<td>15854000</td>
<td>18257000</td>
<td>28257000</td>
</tr>
</tbody>
</table>

Table 366. ECC projective to affine average computation times

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>640</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>47600</td>
<td>78000</td>
<td>148300</td>
<td>253000</td>
<td>419000</td>
<td>838400</td>
<td>1049300</td>
</tr>
</tbody>
</table>

Table 367. ECC complete addition average computation times

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>640</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10000</td>
<td>12000</td>
<td>18000</td>
<td>26000</td>
<td>39000</td>
<td>53000</td>
<td>89000</td>
</tr>
</tbody>
</table>

Table 368. Point on elliptic curve Fp check average computation times

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>521</th>
<th>640</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3400</td>
<td>4200</td>
<td>6100</td>
<td>8300</td>
<td>10900</td>
<td>17200</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
There are four individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. PKA unsupported operation error (OPERRF), see Section 36.3.7
2. Access to unmapped address (ADDRERRF), see Section 36.3.7
3. PKA RAM access while PKA operation is in progress (RAMERRF), see Section 36.3.7
4. PKA end of operation (PROCENDF)

The interrupt sources are connected to the same global interrupt request signal pka_it.

The user can enable or disable above interrupt sources individually by changing the mask bits in the PKA control register (PKA_CR). Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

Table 370 gives a summary of the available features.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKA</td>
<td>Unsupported operation</td>
<td>OPERRF</td>
<td>OPERRIE</td>
<td>Set OPERRFC bit</td>
</tr>
<tr>
<td></td>
<td>Access to unmapped address</td>
<td>ADDRERRF</td>
<td>ADDRERRIE</td>
<td>Set ADDRERRFC bit</td>
</tr>
<tr>
<td></td>
<td>PKA RAM access error</td>
<td>RAMERRF</td>
<td>RAMERRIE</td>
<td>Set RAMERRFC bit</td>
</tr>
<tr>
<td></td>
<td>PKA end of operation</td>
<td>PROCENDF</td>
<td>PROCENDIE</td>
<td>Set PROCENDFC bit</td>
</tr>
</tbody>
</table>

Table 369. Montgomery parameters average computation times

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>3072</th>
<th>4096</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8600</td>
<td>8710</td>
<td>11870</td>
<td>17000</td>
<td>102000</td>
<td>410000</td>
<td>506000</td>
<td>822000</td>
</tr>
</tbody>
</table>

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.
36.7 PKA registers

36.7.1 PKA control register (PKA_CR)

Address offset: 0x00
Reset value: 0x0000 0000

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

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

Bit 21 **OPERRIE**: Operation error interrupt enable
- 0: No interrupt is generated when OPERRF flag is set in PKA_SR.
- 1: An interrupt is generated when OPERRF flag is set in PKA_SR.

Bit 20 **ADDRERRIE**: Address error interrupt enable
- 0: No interrupt is generated when ADDRERRF flag is set in PKA_SR.
- 1: An interrupt is generated when ADDRERRF flag is set in PKA_SR.

Bit 19 **RAMERRIE**: RAM error interrupt enable
- 0: No interrupt is generated when RAMERRF flag is set in PKA_SR.
- 1: An interrupt is generated when RAMERRF flag is set in PKA_SR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDIE**: End of operation interrupt enable
- 0: No interrupt is generated when PROCENDF flag is set in PKA_SR.
- 1: An interrupt is generated when PROCENDF flag is set in PKA_SR.

Bits 16:14 Reserved, must be kept at reset value.
Bits 13:8 **MODE[5:0]**: PKA operation code

- 000000: Montgomery parameter computation then modular exponentiation
- 000001: Montgomery parameter computation only
- 000010: Modular exponentiation only (Montgomery parameter must be loaded first)
- 000011: Modular exponentiation (protected, used when manipulating secrets)
- 100000: Montgomery parameter computation then ECC scalar multiplication (protected)
- 100100: ECDSA sign (protected)
- 100110: ECDSA verification
- 101000: Point on elliptic curve Fp check
- 000111: RSA CRT exponentiation
- 001000: Modular inversion
- 001001: Arithmetic addition
- 001010: Arithmetic subtraction
- 001011: Arithmetic multiplication
- 001100: Arithmetic comparison
- 001101: Modular reduction
- 001110: Modular addition
- 001111: Modular subtraction
- 010000: Montgomery multiplication
- 100011: ECC complete addition
- 100111: ECC double base ladder
- 101111: ECC projective to affine

When an operation not listed here is written by the application with EN bit set, OPERRF bit is set in PKA_SR register, and the write to MODE bitfield is ignored. When PKA is configured in limited mode (LMF = 1 in PKA_SR), writing a MODE different from 0x26 with EN bit to 1 triggers OPERRF bit to be set and write to MODE bit is ignored.

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

**Bit 1** START: start the operation

Writing 1 to this bit starts the operation which is selected by MODE[5:0], using the operands and data already written to the PKA RAM. This bit is always read as 0.

When an illegal operation is selected while START bit is set no operation is started, and OPERRF bit is set in PKA_SR.

*Note: START is ignored if PKA is busy.*

**Bit 0** EN: PKA enable

- 0: Disable PKA
- 1: Enable PKA. PKA becomes functional when INITOK is set by hardware in PKA_SR.

When an illegal operation is selected while EN = 1, OPERRF bit is set in PKA_SR. See PKA_CR.MODE bitfield for details.

*Note: When EN = 0, PKA RAM can still be accessed by the application.*
### 36.7.2 PKA status register (PKA_SR)

Address offset: 0x04
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPERRF</td>
<td>Operation error flag</td>
</tr>
<tr>
<td>ADDRERRF</td>
<td>Address error flag</td>
</tr>
<tr>
<td>RAMERRF</td>
<td>PKA RAM error flag</td>
</tr>
<tr>
<td>PROCENDF</td>
<td>PKA end of operation flag</td>
</tr>
<tr>
<td>BUSY</td>
<td>Busy flag</td>
</tr>
<tr>
<td>LMF</td>
<td>Limited mode flag</td>
</tr>
</tbody>
</table>

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

- **Bit 21** **OPERRF**: Operation error flag
  - 0: No event error
  - 1: An illegal or unknown operation has been selected in PKA_CR register

- **Bit 20** **ADDRERRF**: Address error flag
  - 0: No address error
  - 1: Address access is out of range (unmapped address)

- **Bit 19** **RAMERRF**: PKA RAM error flag
  - 0: No PKA RAM access error
  - 1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).

- **Bit 18** Reserved, must be kept at reset value.

- **Bit 17** **PROCENDF**: PKA end of operation flag
  - 0: Operation in progress
  - 1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.

- **Bit 16** **BUSY**: Busy flag
  - 0: No operation is in progress (default)
  - 1: An operation is in progress

  If PKA is started with a wrong opcode, it stays busy for a couple of cycles, then it aborts automatically the operation and goes back to ready (BUSY = 0).

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

- **Bit 1** **LMF**: Limited mode flag
  - 0: All values documented in MODE bitfield can be used.
  - 1: Only ECDSA verification (MODE = 0x26) is supported by the PKA.
36.7.3 **PKA clear flag register (PKA_CLRFR)**

Address offset: 0x08
Reset value: 0x0000 0000

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

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

Bit 21 **OPERRFC**: Clear operation error flag
0: No action
1: Clear the OPERRF flag in PKA_SR

Bit 20 **ADDRERRFC**: Clear address error flag
0: No action
1: Clear the ADDRERRF flag in PKA_SR

Bit 19 **RAMERRFC**: Clear PKA RAM error flag
0: No action
1: Clear the RAMERRF flag in PKA_SR

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDFC**: Clear PKA end of operation flag
0: No action
1: Clear the PROCENDF flag in PKA_SR

Bits 16:0 Reserved, must be kept at reset value.

**Note:** Reading PKA_CLRFR returns all 0s.

36.7.4 **PKA RAM**

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA_AHB interface.

RAM size is 5336 bytes (max word offset: 0x14D0)

**Note:** PKA RAM cannot be used just after a PKA reset or a product reset, as described in Section 36.3.3: PKA reset and clocks.
### PKA register map

#### Table 371. PKA register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | PKA_CR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x004  | PKA_SR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x008  | PKA_CLRFR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |

Refer to **Section 2.3 on page 111** for the register boundary addresses.
37 On-the-fly decryption engine (OTFDEC)

37.1 Introduction

OTFDEC allows on-the-fly decryption of the AHB traffic based on the read request address information. Four independent and non-overlapping encrypted regions can be defined in OTFDEC.

OTFDEC uses AES-128 in counter mode to achieve the lowest possible latency. As a consequence, each time the content of one encrypted region is changed, the entire region must be re-encrypted with a different cryptographic context (key or initialization vector). This constraint makes OTFDEC suitable to decrypt read-only data or code, stored in external NOR flash.

Note: When OTFDEC is used in conjunction with OCTOSPI, it is mandatory to access the flash memory using the memory-mapped mode of the flash memory controller.

When security is enabled in the product, OTFDEC can be programmed only by a secure host.

37.2 OTFDEC main features

- On-the-fly 128-bit decryption during the OCTOSPI Memory-mapped read operations (single or multiple).
  - Use of AES in counter (CTR) mode, with two 128-bit keystream buffers
  - Support for any read size
  - Physical address of the reads used for the encryption/decryption
- Up to four independent encrypted regions
  - Granularity of the region definition: 4096 bytes
  - Region configuration write-locking mechanism
  - Each region has its own 128-bit key, two bytes firmware version, and eight bytes application-defined nonce. At least one of those must be changed each time an encryption is performed by the application.
- Encryption keys confidentiality and integrity protection
  - Write-only registers, with software locking mechanism
  - Availability of 8-bit CRC as public key information
- Support for OCTOSPI pre-fetching mechanism
- Possibility to select an enhanced encryption mode to add a proprietary layer of protection on top of AES stream cipher (execute only)
- Privileged-aware AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- Secure only programming if TrustZone security is enabled in the product
- Encryption mode
37.3 OTFDEC functional description

37.3.1 OTFDEC block diagram

Figure 333. OTFDEC block diagram

1. otfdec_tzen

37.3.2 OTFDEC internal signals

Table 372 describes a list of useful to know internal signals available at OTFDEC level, not at the product level (on pads).

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>otfdec_hclk</td>
<td>Digital input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>otfdec_it</td>
<td>Digital output</td>
<td>OTFDEC global interrupt request</td>
</tr>
<tr>
<td>otfdec_tzen</td>
<td>Digital input</td>
<td>OTFDEC TrustZone enable, controlling TrustZone features of the peripheral (TZEN)</td>
</tr>
</tbody>
</table>
On-the-fly decryption engine (OTFDEC) RM0481

The TZEN option bit in FLASH is used to activate TrustZone in the device.
- TZEN = 1: TrustZone security is enabled in the product.
- TZEN = 0: TrustZone security is disabled in the product.

37.3.3 OTFDEC on-the-fly decryption

Introduction

Typical usage for OTFDEC is shown on Figure 334.

Figure 334. Typical OTFDEC use in a SoC

Original purpose of OTFDEC is to protect the confidentiality of read-only firmware libraries stored in external SPI NOR flash devices.

A special locking scheme is available in OTFDEC in order to protect the integrity of the decryption keys and also to protect the other configurations against software denial of services attacks. OTFDEC access to most registers can be made privileged-only by setting PRIV bit in OTFDEC_PRIVCFGR register. OTFDEC is only writeable by TrustZone CPU, when TrustZone security is activated.

When OTFDEC is used in conjunction with OCTOSPI, it is mandatory to read the flash memory using the Memory-mapped mode of the flash controller.

On top of decrypting on-the-fly, OTFDEC can also encrypt 32-bit word at a time (see Section 37.5.3: Encrypting for OTFDEC for more details).

OTFDEC architecture

OTFDEC analyzes all AHB read transfers on the associated AHB bus. If the read request is within one of the four regions programmed in OTFDEC, the control logic triggers a keystream computation based on AES algorithm in counter mode. This keystream is then used to decrypt on-the-fly the data present in the read transfer from the OCTOSPI AHB master, tying low the HREADYOUT signal of this master while the keystream information is being computed (this takes up to 11 cycles). Any accesses outside the enabled OTFDEC regions belong to a non-encrypted region.

Each OTFDEC region is programmed through OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR, OTFDEC_RxNONCER and
OTFDEC_RxKEYR registers, where x = 1 to 4. In OTFDEC_RxCFGR, the MODE bits define the OTFDEC operating mode (standard or enhanced encryption).

Granularity for the region determination is 4096 bytes.

**Note:** Although OTFDEC does not prevent region overlapping, it is not a valid programming and it must be avoided by application software.

OTFDEC can decrypt incremental or wrap bursts only if they do not cross the 4096-byte aligned address boundaries.

### 37.3.4 OTFDEC usage of AES in counter mode decryption

Figure 335 shows how OTFDEC uses industry standard Advanced Encryption Standard (AES) algorithm in counter chaining mode. This mode is specified by NIST in *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

**Figure 335. AES CTR decryption flow**

Every 128-bit data block, a special keystream information is computed using AES block cipher, as defined below:

- initialization vector \( AES\_IV[127:0] = RxNONCER1[31:0] || RxNONCER0[31:0] || 0b0000 0000 0000 0000 || RxCFGR[31:16] || 0b00 || (x-1) || ReadAddress[31:4] \)
- key material \( AES\_KEY[127:0] = RxKEYR3[31:0] || RxKEYR2[31:0] || RxKEYR1[31:0] || RxKEYR0[31:0] \)

**Note:** Above x is the RegionID of the selected encrypted region (x=1 to 4), ReadAddress is the AHB address of the encrypted data block, modulo 128-bit.

Resulting 128-bit keystream is XORed with 128-bit cipher text data to produce the 128-bit clear text data.

- \( AES\_DIN \) and \( AES\_DOUT \) data blocks are constructed following the rule below ("||" represents a binary concatenation):
  \[
  AES\_Dx[127:0] = AHB\_word(@ | 0xC)[31:0] || AHB\_word(@ | 0x8)[31:0] || AHB\_word(@ | 0x4)[31:0] || AHB\_word(@ | 0x0)[31:0], \]
  where @ is the hexadecimal address used to compute the keystream (ReadAddress[31:4] above).
When the read request is not within an encrypted region, or the decryption is not enabled in this region, the AHB data is not changed.

Note: When the application sets the MODE bitfield to 11 in OTFDEC_RxCFGR, an additional layer of protection is added on top of the AES stream cipher. This enhanced encryption mode can only be used with instructions (execute-only region).

37.3.5 Flow control management

Figure 336 shows how OTFDEC manages one INCR4 AHB burst that corresponds to one 128-bit AES data block.

Figure 336. OTFDEC flow control overview (dual burst read request)

with the following notes:
1. OTFDEC enforces HREADY signal from the AHB master low as it is not ready to decrypt data (keystream computation).
2. Thanks to the keystream buffer, OTFDEC can be ready to process a new batch of data within 12 cycles in this configuration (120 MHz AHB clock, 104 MHz SPI bus delivering 2 bytes per SPI clock).

37.3.6 OTFDEC error management

OTFDEC automatically manages errors defined as below:
- Illegal read to OTFDEC_RxKEYR registers
- Illegal write to OTFDEC_RxKEYR registers while CONFIGLOCK or KEYLOCK = 1 in OTFDEC_RxCFGR, while the access is secure. If the security is disabled in the product, the same error occurs when the access is nonsecure.
- Illegal write to OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR or OTFDEC_RxNONCER registers while CONFIGLOCK = 1 in OTFDEC_RxCFGR (x = 1 to 4), while the access is secure. If the security is disabled in the product the same error occurs when the access is nonsecure.
- Illegal read to an execute-only region (MODE[1:0] = 11). Such illegal request returns 0x0, without bus error.
- Execution request to a region while encryption is enabled (ENC = 1). The request returns 0x0, without bus error.
- Key error: read request to an encrypted region while its key registers are null or not properly initialized (KEYCRC = 0x0). Source of the error can be an incorrect key loading sequence (see KEYCRC in OTFDEC_RxCFGR) or it can be an abort event.
(tamper detection, unauthorized debug connection, untrusted boot, RDP level regression). Such read request returns 0x0, without bus error.

- Write to any registers while the access is nonsecure, if TrustZone security is enabled in the product.

  This last error is managed and cleared through TrustZone interrupt controller, as described in the GTZC section of the product reference manual.

For these errors (except the last one), an interrupt can be generated if the SEIE, XONEIE or KEIE bit is set in OTFDEC_IER register (see Section 37.4).

**Note:** After a key error, OTFDEC keys must be properly initialized again, and a reset of OTFDEC may be needed if registers are locked.

### 37.4 OTFDEC interrupts

There are three independent maskable interrupt sources generated by the OTFDEC, signaling following security events:

- Illegal read or write access to keys (SEIF flag), see Section 37.3.6
- Illegal write to a region configuration while CONFIGLOCK = 1 (SEIF flag), see Section 37.3.6
- Read access to an execute-only region (MODE[1:0] = 11), triggering the XONEIF flag
- Executing while encryption is enabled (XONEIF flag)
- Key error (encrypted regions read as zero) triggering the KEIF flag, see Section 37.3.6.

Interrupt sources are connected to the same global interrupt request signal.

OTFDEC interrupt sources can be enabled/disabled by setting the corresponding SEIE, XONEIE or KEIE bit in OTFDEC_IER, as described in Table 373. Status of the interrupt event is found in OTFDEC_ISR, and this event can be cleared using OTFDEC_ICR.

### Table 373. OTFDEC interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag(1)</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTFDEC</td>
<td>Security error</td>
<td>SEIF</td>
<td>SEIE</td>
<td>Set SEIF in OTFDEC_ICR</td>
</tr>
<tr>
<td></td>
<td>Execute-only</td>
<td>XONEIF</td>
<td>XONEIE</td>
<td>Set XONEIF in OTFDEC_ICR</td>
</tr>
<tr>
<td></td>
<td>Execute while encryption</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Key error</td>
<td>KEIF</td>
<td>KEIE</td>
<td>Set KEIF in OTFDEC_ICR</td>
</tr>
</tbody>
</table>

1. The event flags are found in the OTFDEC_ISR register.

### 37.5 OTFDEC application information

#### 37.5.1 OTFDEC initialization process

**Introduction**

One key aspect of OTFDEC is the trusted initialization of its registers, as it involves secret keys. Two trusted initialization schemes are recommended here below.
Note: Those sequences are for production code, as during firmware development, it is not always recommended to lock the key or the region configuration. Writes to configuration registers are effective when the configuration locks allow it, even if the region is enabled.

Initialization scheme 1: one key for all regions

In this scheme, one entity owns the secret key used to decrypt the four protected regions. The recommended OTFDEC configuration sequence is described below:

1. For \( x = 1 \) to \( 4 \), write the correct \( \text{MODE}[1:0] \) value in OTFDEC\_RxCFGFR.
2. For \( x = 1 \) to \( 4 \), program OTFDEC\_RxKEYR registers using the sequence described in KEYCRC (to have a valid CRC). Warning as key registers are write only.
3. For \( x = 1 \) to \( 4 \), check the key CRC. If OK, set KEYLOCK bit in OTFDEC\_RxCFGFR. This bit cannot be cleared (key registers in this region \( x \) are no more writable).
4. To do to decrypt a region \( x \) (task that does not necessarily have to be performed by the entity that owns the decryption keys):
   a) Verify if the key CRC corresponds to the encrypted binary stored in the region.
   b) Fill the detailed information corresponding to this binary (nonce, start address, end address, version number).
   c) Enable decryption of this region using REG\_EN.
   d) Set CONFIGLOCK bit in OTFDEC\_RxCFGFR. This bit cannot be cleared (the region configuration is no more writable).

Caution: For a given region, when \( \text{MODE} \) bits are changed, the key registers and associated CRC are cleared by hardware. As a consequence, step 1 above must be done before step 2, and \( \text{MODE} \) bits must not be modified after step 2.

Initialization scheme 2: one key per region

In this scheme, one entity can own the secret used to decrypt one (or more) protected region. The recommended OTFDEC configuration sequence is described below:

1. To do to decrypt a region \( x \) (this task must be performed by the entity that owns the corresponding key):
   a) Write the correct \( \text{MODE}[1:0] \) value in OTFDEC\_RxCFGFR.
   b) Program OTFDEC\_RxKEYR registers using the sequence described in KEYCRC (to have a valid CRC). Warning as key registers are write only.
   c) Check the key CRC. If OK, set KEYLOCK bit in OTFDEC\_RxCFGFR. This bit cannot be cleared (key registers are no more writable).
   d) Fill the detailed information corresponding to the protected firmware (nonce, start address, end address, version number).
   e) Enable decryption of this region using REG\_EN.
   f) Set CONFIGLOCK bit in OTFDEC\_RxCFGFR. This bit cannot be cleared (the region configuration is no more writable).

Caution: For a given region, when \( \text{MODE} \) bits are changed, the key registers and associated CRC are cleared by hardware. As a consequence step a) above must be done before step b), and \( \text{MODE} \) bits must not be modified after step b).
37.5.2 OTFDEC and power management

Each time OTFDEC is reset, the correct key loading sequence described in Section 37.5.1 must be performed (in this case KEYCRC = 0 in OTFDEC_RxCFGR).

It is recommended for application software to verify this point each time OTFDEC is reset by hardware.

37.5.3 Encrypting for OTFDEC

Code and data standard encryption

OTFDEC uses standard AES in counter mode when processing a binary stored in a protected region with MODE[1:0] = 10. When this mode is selected, any AES compatible hardware accelerator or library can be used to encrypt those protected libraries. OTFDEC can be used as well, as described in enhanced encryption section below (with MODE[1:0] = 10).

Definition and endianness of the AES inputs and outputs are defined in Section 37.3.4: OTFDEC usage of AES in counter mode decryption.

Enhanced encryption with OTFDEC

OTFDEC uses a proprietary layer of protection on top of the standard AES in counter mode when processing a code stored in a protected region with MODE[1:0] = 11.

Enhanced encryption mode can be used to increase the robustness against tampering.

Recommended sequence to encrypt using OTFDEC is described below:

1. The application in charge of the encryption sets the ENC bit in OTFDEC_CR. This application must run in TrustZone secure mode when TrustZone security is enabled in the product. If PRIV bit is set in OTFDEC_PRIVCFGR, this application must be privileged.

2. Encryption application initializes OTFDEC as described in Section 37.5.1: OTFDEC initialization process. OCTOSPI must also be properly clocked, so that OTFDEC is fully functional in encryption mode. This step can also be done before step 1.

3. Encryption application writes 32-bit of clear-text data at the expected protected address, then reads it back encrypted at the same address to store it in RAM. Note that this data stays inside the device, as it is intercepted by OTFDEC in encryption mode.

4. Encryption application goes back to previous step (changing the address) until the whole binary is processed.

5. Encryption application clears the ENC bit in OTFDEC_CR. Another application can then take the encrypted binary and flash it to the correct address in external flash.

There are few important notes about this procedure:

• Encryption granularity is 32-bit (single 32-bit access is mandatory).

• While ENC bit is set, reads to non-encrypted regions return normal data (such as no encryption nor decryption). While in encryption mode, no access to OCTOSPI (including registers) must be done. This is because the OTFDEC cuts the communication with OCTOSPI while ENC bit is set.

• OTFDEC does not support execution while ENC = 1 (only encrypted data reads). Upon illegal execution detection a XONEIF flag is raised and zero is returned.
37.5.4 OTFDEC key CRC source code

Below is the CRC source code that can be used to compare with the result of the computation provided by OTFDEC in KEYCRC bitfield after loading the keys in OTFDEC_RxKEYR registers.

```c
uint8_t getCRC(uint32_t * keyin)
{
    const uint8_t CRC7_POLY = 0x7;
    const uint32_t key_strobe[4] = {0xAA55AA55, 0x3, 0x18, 0xC0};
    uint8_t  i, j, k, crc = 0x0;
    uint32_t  keyval;

    for (j = 0; j < 4; j++)
    {
        keyval = *(keyin+j);
        if (j == 0)
        {
            keyval ^= key_strobe[0];
        }
        else
        {
            keyval ^= (key_strobe[j] << 24) | (crc << 16) | (key_strobe[j] << 8) | crc;
        }

        for (i = 0, crc = 0; i < 32; i++)
        {
            k = (((crc >> 7) ^ (keyval >> (31-i))&0xF)) & 1;
            crc <<= 1;
            if (k)
            {
                crc ^= CRC7_POLY;
            }
        }
        crc^=0x55;
    }
    return crc;
}
```
# OTFDEC registers

## OTFDEC control register (OTFDEC_CR)

Address offset: 0x0  
Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

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

**Bit 0 ENC:** Encryption mode bit

When this bit is set, OTFDEC is used in encryption mode, during which application can write clear text data then read back encrypted data. When this bit is cleared (default), OTFDEC is used in decryption mode, during which application only read back decrypted data. For both modes, cryptographic context (keys, nonces, firmware versions) must be properly initialized. When this bit is set, only data accesses are allowed (zeros are returned otherwise, and XONEIF is set). When MODE = 11, enhanced encryption mode is automatically selected.

0: OTFDEC working in decryption mode  
1: OTFDEC working in encryption mode

**Note:** When ENC bit is set, no access to OCTOSPI must be done (registers and Memory-mapped region).
### 37.6.2 OTFDEC privileged access control configuration register (OTFDEC_PRIVCFGR)

Address offset: 0x10  
Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Bit 0 PRIV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:1</td>
<td>Reserved, must be kept at reset value.</td>
<td>Bit 0 PRIV</td>
</tr>
<tr>
<td>Bit 0 PRIV</td>
<td>Privileged access protection.</td>
<td>0: No additional protection is added on OTFDEC register accesses.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: An additional protection is added when accessing all registers except OTFDEC_PRIVCFGR:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Unprivileged read accesses to registers return zeros</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Unprivileged write accesses to registers are ignored.</td>
</tr>
<tr>
<td>Note</td>
<td>This bit can only be written in privileged mode. There is no limitations on reads.</td>
<td></td>
</tr>
</tbody>
</table>

### 37.6.3 OTFDEC region x configuration register (OTFDEC_RxCFGR)

Address offset: 0x20 + 0x30 * (x - 1), (x = 1 to 4)  
Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

Writes are ignored if CONFIGLOCK bit is set to 1.
Bits 31:16 **REG_VERSION[15:0]:** region firmware version
   This 16-bit bitfield must be correctly initialized before the region corresponding REG_EN bit is set in OTFDEC_RxCFGR.

Bits 15:8 **KEYCRC[7:0]:** region key 8-bit CRC
   When KEYLOCK = 0, KEYCRC bitfield is automatically computed by hardware while loading the key of this region in this exact sequence: KEYR0 then KEYR1 then KEYR2 then finally KEYR3 (all written once). A new computation starts as soon as a new valid sequence is initiated, and KEYCRC is read as zero until a valid sequence is completed.
   When KEYLOCK = 1, KEYCRC remains unchanged until the next reset.
   CRC computation is an 8-bit checksum using the standard CRC-8-CCITT algorithm $x^8 + x^2 + x + 1$ (according the convention). Source code is available in Section 37.5.4.
   This field is read only.
   Note: CRC information is updated only after the last bit of the key has been written.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MODE[1:0]:** operating mode
   This bitfield selects the OTFDEC operating mode for this region:
   10: All read accesses are decrypted (instruction or data).
   11: Enhanced encryption mode is activated, and only instruction accesses are decrypted
   Others: Reserved
   When MODE $\neq$ 11, the standard AES encryption mode is activated.
   When either of the MODE bits are changed, the region key and associated CRC are zeroed.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **KEYLOCK:** region key lock
   0: Writes to this region KEYRx registers are allowed.
   1: Writes to this region KEYRx registers are ignored until next OTFDEC reset. KEYCRC bitfield is locked.
   Note: This bit is set once: if this bit is set, it can only be reset to 0 if the OTFDEC is reset.

Bit 1 **CONFIGLOCK:** region config lock
   0: Writes to this region OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR and OTFDEC_RxNONCERy registers are allowed.
   1: Writes to this region OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR and OTFDEC_RxNONCERy registers are ignored until next OTFDEC reset.
   Note: This bit is set once. If this bit is set, it can only be reset to 0 if OTFDEC is reset. Setting this bit forces KEYLOCK bit to 1.

Bit 0 **REG_EN:** region on-the-fly decryption enable
   0: On-the-fly decryption is disabled for this region.
   1: On-the-fly decryption is enabled for this region. Data are XORed with the corresponding keystream.
   Note: Garbage is decrypted if region context (version, key, nonce) is not valid when this bit is set.
37.6.4 OTFDEC region x start address register (OTFDEC_RxSTARTADDR)

Address offset: 0x24 + 0x30 * (x - 1), (x = 1 to 4)
Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0 REG_START_ADDR[31:0]: Region AHB start address
This register must be written before the region corresponding REG_EN bit in the OTFDEC_RxCFGR register is set.
Writing to this register is discarded if performed while the region CONFIGLOCK bit in the OTFDEC_RxCFGR register is set.

Note: When determining the region the first 12 bits (LSB) and the last 4 bits (MSB) are ignored.
When this register is accessed in read the 4 MSB bits and the 12 LSB bits return zeros.

37.6.5 OTFDEC region x end address register (OTFDEC_RxENDADDR)

Address offset: 0x28 + 0x30 * (x - 1), (x = 1 to 4)
Reset value: 0x0000 0FFF

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

<table>
<thead>
<tr>
<th>REG_END_ADDR[15:0]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
37.6.6 OTFDEC region x nonce register 0 (OTFDEC_RxNONCER0)

Address offset: 0x2C + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0 REG_NONCE[31:0]: Region nonce, bits [31:0]

This register must be written before the region corresponding REG_EN bit in OTFDEC_RxCFGR is set.

Writing is discarded in this register if performed while the region CONFIGLOCK bit in the OTFDEC_RxCFGR is set.
37.6.7 OTFDEC region x nonce register 1 (OTFDEC_RxNONCER1)

Address offset: 0x30 + 0x30 * (x - 1), (x = 1 to 4)
Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

 Bits 31:0 REG_NONCE[63:32]: Region nonce, bits [63:32]
Refer to the OTFDEC_RxNONCER0 register for description of the NONCE[63:0] bitfield.

37.6.8 OTFDEC region x key register 0 (OTFDEC_RxKEYR0)

Address offset: 0x34 + 0x30 * (x - 1), (x = 1 to 4)
Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

 Bits 31:0 REG_KEY[31:16]: Region key, bits [31:16]
This register must be written before the region corresponding REG_EN bit in OTFDEC_RxCFGR is set.
Reading this register returns a zero value. Writing to this register is discarded if performed while the region CONFIGLOCK or KEYLOCK bit is set in the OTFDEC_RxCFGR.
Note: When application successfully changes MODE bits in OTFDEC_RxCFGR and OTFDEC_RxKEYR, and associated KEYCRC are erased.
37.6.9 **OTFDEC region x key register 1 (OTFDEC_RxKEYR1)**

Address offset: 0x38 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

**REG_KEY[95:64]**

Bits 31:0 **REG_KEY[63:32]**: Region key, bits [63:32]

Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

37.6.10 **OTFDEC region x key register 2 (OTFDEC_RxKEYR2)**

Address offset: 0x3C + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

**REG_KEY[95:64]**

Bits 31:0 **REG_KEY[95:64]**: Region key, bits [95:64]

Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.
37.6.11  **OTFDEC region x key register 3 (OTFDEC_RxKEYR3)**

Address offset: 0x40 + 0x30 * (x - 1), (x = 1 to 4)
Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0  **REG_KEY[127:96]**: Region key, bits [127:96]

Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

37.6.12  **OTFDEC interrupt status register (OTFDEC_ISR)**

Address offset: 0x300
Reset value: 0x0000 0000

Unprivileged reads return zero if PRIV bit is set in OTFDEC_PRIVCFGR.

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

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

Bit 2  **KEIF**: Key error interrupt flag status

This bit is set by hardware and read only by application. The bit is set when a read access occurs on an encrypted region, while its key registers is null or not properly initialized (KEYCRC = 0x0).

This bit is cleared when the application sets in OTFDEC_ICR the corresponding bit to 1.

0: OTFDEC operates properly.
1: Read access detected on an enabled encrypted region with its key registers null or not properly initialized (KEYCRC = 0x0). OTFDEC returns a zeroed value for the read, and an optional interrupt is generated if bit KEIE is set to 1 in OTFDEC_IER.

After KEIF is set any subsequent read to the region with bad key registers returns a zeroed value. This state remains until those key registers are properly initialized (KEYCRC not zero).
Bit 1 **XONEIF**: Execute-only execute-never error interrupt flag status

This bit is set by hardware and read only by application. This bit is set when a read access and not an instruction fetch is detected on any encrypted region with MODE bits set to 11. Lastly, XONEIF is also set when an execute access is detected while encryption mode is enabled.
This bit is cleared when application sets in OTFDEC_ICR the corresponding bit to 1.
0: No execute-only error status. No interrupt pending.
1: Read access detected on one region with MODE bits set to 11 or execute access detected while ENC = 1. OTFDEC returns a zeroed value for the illegal access, and an optional interrupt is generated if bit XONEIE is set to 1 in OTFDEC_IER.

Bit 0 **SEIF**: Security error interrupt flag status

This bit is set by hardware and read only by application. This bit is set when at least one security error has been detected.
This bit is cleared when application sets in OTFDEC_ICR the corresponding bit to 1.
0: No security error status. No interrupt pending.
1: Security error flag status, with interrupt pending. Actual interrupt generation is dependent on OTFDEC_IER corresponding bit SEIE.

### 37.6.13 OTFDEC interrupt clear register (OTFDEC_ICR)

Address offset: 0x304
Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:3 Reserved, must be kept at reset value.
Bit 2 **KEIF:** Key error interrupt flag clear
This bit is written by application, and always read as 0.
0: KEIF flag status is not affected.
1: KEIF flag status is cleared in OTFDEC_ISR.

*Note:* Clearing KEIF does not solve the source of the problem (bad key registers). To be able to access again any encrypted region, OTFDEC key registers must be properly initialized again.

Bit 1 **XONEIF:** Execute-only execute-never error interrupt flag clear
This bit is written by application, and always read as 0.
0: XONEIF flag status is not affected.
1: XONEIF flag status is cleared in OTFDEC_ISR.

Bit 0 **SEIF:** Security error interrupt flag clear
This bit is written by application, and always read as 0.
0: SEIF flag status is not affected.
1: SEIF flag status is cleared in OTFDEC_ISR.

### 37.6.14 OTFDEC interrupt enable register (OTFDEC_IER)

Address offset: 0x308
Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

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

Bits 31:3 Reserved, must be kept at reset value.
Bit 2 **KEIE**: Key error interrupt enable
This bit is read and written by application. It controls the OTFDEC interrupt generation when KEIF flag status is set.
0: Interrupt generation on key error flag KEIF is disabled (masked).
1: Interrupt generation on key error flag KEIF is enabled (not masked).

Bit 1 **XONEIE**: Execute-only execute-never error interrupt enable
This bit is read and written by application. It controls the OTFDEC interrupt generation when XONEIF flag status is set.
0: Interrupt generation on execute-only error XONEIF is disabled (masked).
1: Interrupt generation on execute-only error XONEIF is enabled (not masked).

Bit 0 **SEIE**: Security error interrupt enable
This bit is read and written by application. It controls the OTFDEC interrupt generation when SEIF flag status is set.
0: Interrupt generation on security error SEIF is disabled (masked).
1: Interrupt generation on security error SEIF is enabled (not masked).

### 37.6.15 OTFDEC register map

**Table 374. OTFDEC register map and reset values**

| Offset | Register name                      | REG1_VERSION[15:0] | KEYCRC[7:0] | MDF[1:0] | KELOCK | CFGLOCK | CONFIGLOCK | REG_EN | ENC | 0x00-00C | PRIV | 0x10-01C | Reserved | 0x20-02C | 0x24-028 | 0x2C-030 | 0x30-034 |
|--------|-----------------------------------|--------------------|-------------|----------|--------|---------|------------|-------|-----|----------|------|----------|----------|---------|---------|---------|---------|---------|
| 0x00   | OTFDEC_CR                         |                    |             |          |        |         |            |       |     |          |      |          |          |         |         |         |         |         |
|        | Reset value                       |                    |             |          |        |         |            |       |     |          |      |          |          |         |         |         |         |         |
| 0x04-  | Reserved                          |                    |             |          |        |         |            |       |     |          |      |          |          |         |         |         |         |         |
| 0x0C   |                                   |                    |             |          |        |         |            |       |     |          |      |          |          |         |         |         |         |         |
| 0x10   | OTFDEC_PRIVCFGR                   |                    |             |          |        |         |            |       |     |          |      |          |          |         |         |         |         |         |
|        | Reset value                       |                    |             |          |        |         |            |       |     |          |      |          |          |         |         |         |         |         |
| 0x14-  | Reserved                          |                    |             |          |        |         |            |       |     |          |      |          |          |         |         |         |         |         |
| 0x1C   |                                   |                    |             |          |        |         |            |       |     |          |      |          |          |         |         |         |         |         |
| 0x20   | OTFDEC_R1CFGR1                    | REG1_VERSION[15:0] |            |          |        |         |            |       |     |          |      |          |          |         |         |         |         |         |
|        | Reset value                       | 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x38</td>
<td>OTFDEC_R1KEYR1</td>
<td>REG1_KEY[63:32]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x3C</td>
<td>OTFDEC_R1KEYR2</td>
<td>REG1_KEY[95:64]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x40</td>
<td>OTFDEC_R1KEYR3</td>
<td>REG1_KEY[95:64]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x44</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x50</td>
<td>OTFDEC_R2CFGFR</td>
<td>REG2_VERSION[15:0]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x54</td>
<td>OTFDEC_R2STARTADDR</td>
<td>REG2_VERSION[15:0]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x58</td>
<td>OTFDEC_R2NONCER0</td>
<td>REG2_NONCE[31:0]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x5C</td>
<td>OTFDEC_R2NONCER1</td>
<td>REG2_NONCE[63:32]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x60</td>
<td>OTFDEC_R2KEYR0</td>
<td>REG2_KEY[31:0]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x64</td>
<td>OTFDEC_R2KEYR1</td>
<td>REG2_KEY[63:32]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x68</td>
<td>OTFDEC_R2KEYR2</td>
<td>REG2_KEY[95:64]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x6C</td>
<td>OTFDEC_R2KEYR3</td>
<td>REG2_KEY[95:64]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x70</td>
<td>OTFDEC_R3KEYR3</td>
<td>REG3_KEY[64:32]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x74</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x7C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x80</td>
<td>OTFDEC_R3CFGFR</td>
<td>REG3_VERSION[15:0]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x84</td>
<td>OTFDEC_R3STARTADDR</td>
<td>REG3_VERSION[15:0]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x88</td>
<td>OTFDEC_R3ENDADDR</td>
<td>REG3_VERSION[15:0]</td>
<td>00000000000000000000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>Offset Value</td>
<td>Reset Value</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>0x8C</td>
<td>REG3_NONCE[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x90</td>
<td>REG3_NONCE[63:32]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x94</td>
<td>REG3_KEY[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x98</td>
<td>REG3_KEY[63:32]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x9C</td>
<td>REG3_KEY[95:64]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xA0</td>
<td>REG3_KEY[95:64]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xA4</td>
<td>REG4_NONCE[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xA8</td>
<td>REG4_NONCE[63:32]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xAC</td>
<td>REG4_NONCE[95:64]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xB0</td>
<td>REG4_START_ADDR[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xB4</td>
<td>REG4_END_ADDR[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xB8</td>
<td>REG4_NONCE[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xBC</td>
<td>REG4_NONCE[63:32]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xC0</td>
<td>REG4_KEY[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xC4</td>
<td>REG4_KEY[63:32]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xC8</td>
<td>REG4_KEY[95:64]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xCC</td>
<td>REG4_KEY[95:64]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xD0</td>
<td>REG4_KEY[95:64]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x0D4-0x2FC</td>
<td>Reserved</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
</tbody>
</table>
Refer to Section 2.3 for the register boundary addresses.

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x300  | OTFDEC_ISR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset  | value        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x304  | OTFDEC_ICR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset  | value        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x308  | OTFDEC_IER   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset  | value        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Table 374. OTFDEC register map and reset values (continued)
Advanced-control timers (TIM1/TIM8)

TIM1/TIM8 introduction

The advanced-control timers (TIM1/TIM8) consist of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1/TIM8) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in Section 38.3.30: Timer synchronization.

TIM1/TIM8 main features

TIM1/TIM8 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 6 independent channels for:
  - Input capture (but channels 5 and 6)
  - Output compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and Hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management
38.3 TIM1/TIM8 functional description

38.3.1 Block diagram

Figure 337. Advanced-control timer block diagram

Notes:

- **R** Preload registers transferred to active registers on UEV event according to control bit

1. This feature is not available on all timers, refer to Section 38.3.2: TIM1/TIM8 pins and internal signals.
2. See Figure 384: Break and Break2 circuitry overview for details.
38.3.2 TIM1/TIM8 pins and internal signals

The tables in this section summarize the TIM inputs and outputs.

### Table 375. TIM input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_CH1</td>
<td>Input/Output</td>
<td>Timer multi-purpose channels. Each channel can be used for capture, compare or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.</td>
</tr>
<tr>
<td>TIM_CH2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH1N</td>
<td>Output</td>
<td>Timer complementary outputs, derived from TIM_CHx outputs with the possibility to have deadtime insertion.</td>
</tr>
<tr>
<td>TIM_CH2N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH3N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH4N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_ETR</td>
<td>Input</td>
<td>External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.</td>
</tr>
<tr>
<td>TIM_BKIN</td>
<td></td>
<td>Break and Break2 inputs. These inputs can also be configured in bidirectional mode.</td>
</tr>
<tr>
<td>TIM_BKIN2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 376. TIM internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti1_in[15:0]</td>
<td>Input</td>
<td>Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.</td>
</tr>
<tr>
<td>tim_ti2_in[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_ti3_in[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_ti4_in[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_etr[15:0]</td>
<td>Input</td>
<td>External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulsewidth control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.</td>
</tr>
<tr>
<td>tim_itr[15:0]</td>
<td>Input</td>
<td>Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_trgo/tim_trgo2</td>
<td>Output</td>
<td>Internal trigger outputs. These triggers are used by other timers and /or other peripherals.</td>
</tr>
<tr>
<td>Internal signal name</td>
<td>Signal type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>tim_ocref_clr[7:0]</td>
<td>Input</td>
<td>Timer tim_ocref_clr input bus. These inputs can be used to clear the tim ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.</td>
</tr>
<tr>
<td>tim_brk_cmp[8:1]</td>
<td>Input</td>
<td>Break input for internal signals</td>
</tr>
<tr>
<td>tim_brk2_cmp[8:1]</td>
<td>Input</td>
<td>Break2 input for internal signals</td>
</tr>
<tr>
<td>tim_sys_brk[n:0]</td>
<td>Input</td>
<td>System break input. This input gathers the MCU’s system level errors.</td>
</tr>
<tr>
<td>tim_pclk</td>
<td>Input</td>
<td>Timer APB clock</td>
</tr>
<tr>
<td>tim_ker_ck</td>
<td>Input</td>
<td>Timer kernel clock</td>
</tr>
<tr>
<td>tim_cc_it</td>
<td>Output</td>
<td>Timer capture/compare interrupt</td>
</tr>
<tr>
<td>tim_upd_it</td>
<td>Output</td>
<td>Timer update event interrupt</td>
</tr>
<tr>
<td>tim_brk_terr ierr it</td>
<td>Output</td>
<td>Timer break, break2, transition error and index error interrupt</td>
</tr>
<tr>
<td>tim_trg com dir idx it</td>
<td>Output</td>
<td>Timer trigger, commutation, direction and index interrupt</td>
</tr>
<tr>
<td>tim_cc1_dma</td>
<td>Output</td>
<td>Timer capture / compare 1..4 dma requests</td>
</tr>
<tr>
<td>tim_cc2_dma</td>
<td>Output</td>
<td>Timer update dma request</td>
</tr>
<tr>
<td>tim_cc3_dma</td>
<td>Output</td>
<td>Timer trigger dma request</td>
</tr>
<tr>
<td>tim_cc4_dma</td>
<td>Output</td>
<td>Timer commutation dma request</td>
</tr>
</tbody>
</table>
Table 377, Table 378, Table 379 and Table 380 list the sources connected to the tim_ti[4:1] input multiplexers.

### Table 377. Interconnect to the tim_ti1 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti1 inputs</th>
<th>Sources</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM1</td>
<td>TIM8</td>
<td></td>
</tr>
<tr>
<td>tim_ti1_in0</td>
<td>TIM1_CH1</td>
<td>TIM8_CH1</td>
<td></td>
</tr>
<tr>
<td>tim_ti1_in[15:1]</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

### Table 378. Interconnect to the tim_ti2 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti2 inputs</th>
<th>Sources</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM1</td>
<td>TIM8</td>
<td></td>
</tr>
<tr>
<td>tim_ti2_in0</td>
<td>TIM1_CH2</td>
<td>TIM8_CH2</td>
<td></td>
</tr>
<tr>
<td>tim_ti2_in[15:1]</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

### Table 379. Interconnect to the tim_ti3 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti3 inputs</th>
<th>Sources</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM1</td>
<td>TIM8</td>
<td></td>
</tr>
<tr>
<td>tim_ti3_in0</td>
<td>TIM1_CH3</td>
<td>TIM8_CH3</td>
<td></td>
</tr>
<tr>
<td>tim_ti3_in[15:1]</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

### Table 380. Interconnect to the tim_ti4 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti4 inputs</th>
<th>Sources</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM1</td>
<td>TIM8</td>
<td></td>
</tr>
<tr>
<td>tim_ti4_in0</td>
<td>TIM1_CH4</td>
<td>TIM8_CH4</td>
<td></td>
</tr>
<tr>
<td>tim_ti4_in[15:1]</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

Table 381 lists the internal sources connected to the tim_itr input multiplexer.

### Table 381. Internal trigger connection

<table>
<thead>
<tr>
<th>Timer internal trigger input signal</th>
<th>TIM1</th>
<th>TIM8</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_itr0</td>
<td>Reserved</td>
<td>tim1_trgo</td>
</tr>
<tr>
<td>tim_itr1</td>
<td>tim2_trgo</td>
<td>tim2_trgo</td>
</tr>
<tr>
<td>tim_itr2</td>
<td>tim3_trgo</td>
<td>tim3_trgo</td>
</tr>
<tr>
<td>tim_itr3</td>
<td>tim4_trgo</td>
<td>tim4_trgo</td>
</tr>
<tr>
<td>tim_itr4</td>
<td>tim5_trgo</td>
<td>tim5_trgo</td>
</tr>
</tbody>
</table>
Table 381. Internal trigger connection (continued)

<table>
<thead>
<tr>
<th>Timer internal trigger input signal</th>
<th>TIM1</th>
<th>TIM8</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_itr5</td>
<td>tim8_trgo</td>
<td>Reserved</td>
</tr>
<tr>
<td>tim_itr6</td>
<td>tim12_trgo</td>
<td>tim12_trgo</td>
</tr>
<tr>
<td>tim_itr7</td>
<td>tim13_oc1</td>
<td>tim13_oc1</td>
</tr>
<tr>
<td>tim_itr8</td>
<td>tim14_oc1</td>
<td>tim14_oc1</td>
</tr>
<tr>
<td>tim_itr9</td>
<td>tim15_trgo</td>
<td>tim15_trgo</td>
</tr>
<tr>
<td>tim_itr10</td>
<td>tim16_oc1</td>
<td>tim16_oc1</td>
</tr>
<tr>
<td>tim_itr11</td>
<td>tim17_oc1</td>
<td>tim17_oc1</td>
</tr>
<tr>
<td>tim_itr[15:12]</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

*Table 382* lists the internal sources connected to the *tim_etrm* input multiplexer.

**Table 382. Interconnect to the tim_etrm input multiplexer**

<table>
<thead>
<tr>
<th>Timer external trigger input signal</th>
<th>Timer external trigger signals assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM1</td>
</tr>
<tr>
<td>tim_etrm</td>
<td>TIM1_ETR</td>
</tr>
<tr>
<td>tim_etrm[2:1]</td>
<td>Reserved</td>
</tr>
<tr>
<td>tim_etrm3</td>
<td>adc1_awd1</td>
</tr>
<tr>
<td>tim_etrm4</td>
<td>adc1_awd2</td>
</tr>
<tr>
<td>tim_etrm5</td>
<td>adc1_awd3</td>
</tr>
<tr>
<td>tim_etrm[15:6]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

*Table 383, Table 384 and Table 385* list the sources connected to the *tim_brk* and *tim_brk2* inputs.

**Table 383. Timer break interconnect**

<table>
<thead>
<tr>
<th>tim_brk inputs</th>
<th>TIM1</th>
<th>TIM8</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_BKIN</td>
<td>TIM1_BKIN pin</td>
<td>TIM8_BKIN pin</td>
</tr>
<tr>
<td>tim_brk_cmp[8:1]</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Table 384. Timer break2 interconnect**

<table>
<thead>
<tr>
<th>tim_brk2 inputs</th>
<th>TIM1</th>
<th>TIM8</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_BKIN2</td>
<td>TIM1_BKIN2 pin</td>
<td>TIM8_BKIN2 pin</td>
</tr>
<tr>
<td>tim_brk2_cmp[8:1]</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
38.3.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software, even when the counter is running.

The time-base unit includes:
- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

*Note:* The counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler divides the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 338* and *Figure 339* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

<table>
<thead>
<tr>
<th>tim_sys_brk inputs</th>
<th>TIM1/TIM8</th>
<th>Enable bit in SBS_CFG2 register</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_sys_brk0</td>
<td>FLASH double ECC error</td>
<td>ECCL</td>
</tr>
<tr>
<td>tim_sys_brk1</td>
<td>Programmable Voltage Detector (PVD)</td>
<td>PVDL</td>
</tr>
<tr>
<td>tim_sys_brk2</td>
<td>SRAM double ECC error</td>
<td>SEL</td>
</tr>
<tr>
<td>tim_sys_brk3</td>
<td>Cortex®-M33 LOCKUP</td>
<td>CLL</td>
</tr>
<tr>
<td>CSS</td>
<td>Clock Security System</td>
<td>None (always enabled)</td>
</tr>
</tbody>
</table>
Figure 338. Counter timing diagram with prescaler division change from 1 to 2

Figure 339. Counter timing diagram with prescaler division change from 1 to 4
38.3.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):
- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.
Figure 340. Counter timing diagram, internal clock divided by 1

- `tim_psc_ck`
- `CEN`
- `tim_cnt_ck`
- Counter register: 31 32 33 34 35 36 00 01 02 03 04 05 06 07
- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)

Figure 341. Counter timing diagram, internal clock divided by 2

- `tim_psc_ck`
- `CEN`
- `tim_cnt_ck`
- Counter register: 0034 0035 0036 0000 0001 0002 0003
- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)
**Figure 342. Counter timing diagram, internal clock divided by 4**

- `tim_psc_ck`
- `CEN`
- `tim_cnt_ck`

Counter register:
- 0035
- 0036
- 0000
- 0001

- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)

**Figure 343. Counter timing diagram, internal clock divided by N**

- `tim_psc_ck`
- `tim_cnt_ck`

Counter register:
- 1F
- 20
- 00

- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)
Figure 344. Counter timing diagram, update event when ARPE=0
(TIMx_ARR not preloaded)

- **tim_psc_ck**
- **CEN**
- **tim_cnt_ck**
- **Counter register**
  - 31
  - 32
  - 33
  - 34
  - 35
  - 36
  - 00
  - 01
  - 02
  - 03
  - 04
  - 05
  - 06
  - 07
- **Counter overflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**
- **Auto-reload preload register**
  - FF
  - 36

Write a new value in TIMx_ARR
**Downcounting mode**

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn’t change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.
When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

**Figure 346. Counter timing diagram, internal clock divided by 1**
**Figure 347. Counter timing diagram, internal clock divided by 2**

```
<table>
<thead>
<tr>
<th>tim_psc_ck</th>
<th>CEN</th>
<th>tim_cnt_ck</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Counter register:

```
0002 | 0001 | 0000 | 0036 | 0035 | 0034 | 0033 |
```

Counter underflow:

Update event (UEV):

Update interrupt flag (UIF):

---

**Figure 348. Counter timing diagram, internal clock divided by 4**

```
<table>
<thead>
<tr>
<th>tim_psc_ck</th>
<th>CEN</th>
<th>tim_cnt_ck</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Counter register:

```
0001 | 0000 | 0000 | 0001 |
```

Counter underflow:

Update event (UEV):

Update interrupt flag (UIF):
Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-
reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.
1. Here, center-aligned mode 1 is used (for more details refer to Section 38.6: TIM1/TIM8 registers).

Figure 351. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6

Figure 352. Counter timing diagram, internal clock divided by 2
Figure 353. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

Figure 354. Counter timing diagram, internal clock divided by N

Note: Here, center_aligned mode 2 or 3 is updated with an UIF on overflow
Figure 355. Counter timing diagram, update event with ARPE=1 (counter underflow)
38.3.5 Repetition counter

Section 38.3.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode.

Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2\times T_{ck}$, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to Figure 357). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.
In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

**Figure 357. Update rate examples depending on mode and TIMx_RCR register settings**

![Diagram showing update rate examples](image)

**38.3.6 External trigger input**

The timer features an external trigger input tim_etr_in. It can be used as:
- external clock (external clock mode 2, see *Section 38.3.7*)
- trigger for the slave mode (see *Section 38.3.30*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 38.3.9*)

*Figure 358* below describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield. The resulting signal (tim_etrf) is available for three purposes: as an external clock, to condition...
the output (typically to reset a PWM output for a current limitation), and as a trigger for the Slave mode controller.

**Figure 358. External trigger input block**

The `tim_etr_in` input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[3:0] bitfield in the TIMx_AF1 register.

Refer to [Section 38.3.2: TIM1/TIM8 pins and internal signals](#) for the list of sources connected to the `etr_in` input in the product.

### 38.3.7 Clock selection

The counter clock can be provided by the following clock sources:

- **Internal clock** (`tim_ker_ck`)
- **External clock mode1**: external input pin (`tim_ti1` or `tim_ti2`)
- **External clock mode2**: external trigger input (`tim_etr_in`)
- **Encoder mode**

#### Internal clock source (`tim_ker_ck`)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock `tim_ker_ck`.

**Figure 359** shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.
External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

1. Codes ranging from 0100 to 1111 are reserved.
For example, to configure the upcounter to count in response to a rising edge on the `tim_ti2` input, use the following procedure:

1. Configure channel 2 to detect rising edges on the `tim_ti2` input by writing `CC2S = '01'` in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the `IC2F[3:0]` bits in the TIMx_CCMR1 register (if no filter is needed, keep `IC2F=0000`).
3. Select rising edge polarity by writing `CC2P=0` and `CC2NP=0` in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing `SMS=111` in the TIMx_SMCR register.
5. Select `tim_ti2` as the trigger input source by writing `TS=00110` in the TIMx_SMCR register.
6. Enable the counter by writing `CEN=1` in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on `tim_ti2`, the counter counts once and the TIF flag is set. The delay between the rising edge on `tim_ti2` and the actual clock of the counter is due to the resynchronization circuit on `tim_ti2` input.

**Figure 361. Control circuit in external clock mode 1**

![Control circuit in external clock mode 1](image)

**External clock source mode 2**

This mode is selected by writing `ECE=1` in the TIMx_SMCR register. The counter counts at each rising or falling edge on the external trigger input `tim_etr_in`. The Figure 362 gives an overview of the external trigger input block.
Figure 362. External trigger input block

1. Refer to Section 38.3.2: TIM1/TIM8 pins and internal signals.

For example, to configure the upcounter to count each 2 rising edges on `tim_etrin`, use the following procedure:

1. As no filter is needed in this example, write `ETF[3:0]=0000` in the TIMx_SMCR register.
2. Set the prescaler by writing `ETPS[1:0]=01` in the TIMx_SMCR register.
3. Select rising edge detection on the `tim_etrin` input by writing `ETP=0` in the TIMx_SMCR register.
4. Enable external clock mode 2 by writing `ECE=1` in the TIMx_SMCR register.
5. Enable the counter by writing `CEN=1` in the TIMx_CR1 register.

The counter counts once each 2 `tim_etrin` rising edges.

The delay between the rising edge on `tim_etrin` and the actual clock of the counter is due to the resynchronization circuit on the `tim_etrp` signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most $\frac{1}{4}$ of `tim_ker_ck` frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.
38.3.8 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

*Figure 364 to Figure 367* give an overview of one capture/compare channel.

The input stage samples the corresponding TIM_tix input to generate a filtered signal TIM_tixf. Then, an edge detector with polarity selection generates a signal (TIM_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

*Figure 364. Capture/compare channel (example: channel 1 input stage)*
The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

**Figure 365. Capture/compare channel 1 main circuit**
The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.
In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 38.3.9 Input capture mode

In Input capture mode, the capture/compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to ‘0’ or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_t1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the tim_t1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.

- Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let’s imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on tim_t1 when 8 consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

- Select the edge of the active transition on the tim_t1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).

- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).

- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.

- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.
Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

38.3.10 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIMx_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges)

This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tixfp1 input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two tim_tixfp signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulsewidth of a PWM signal applied on tim_t1 can be measured using the following procedure:

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_t1 selected).
- Select the active polarity for tim_t1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_t1 selected).
- Select the active polarity for tim_t1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP=’10’ (active on falling edge).
- Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_t1fp1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.
38.3.11 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_OCCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP=0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

38.3.12 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while channel 5 and 6 are only available inside the microcontroller (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=0000), be set
active (OCxM=0001), be set inactive (OCxM=0010) or can toggle (OCxM=0011) on match.
• Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
• Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIMx_DIER register).
• Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure
1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 369.
38.3.13 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.
**PWM edge-aligned mode**

- **Upcounting configuration**
  
  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the *Upcounting mode on page 1435.*

  In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then tim_ocxref is held at ‘1’. If the compare value is 0 then tim_ocxref is held at ‘0’. *Figure 370* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

**Figure 370. Edge-aligned PWM waveforms (ARR=8)**

<table>
<thead>
<tr>
<th>Counter register</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCRx=4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_ocxref</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx=8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_ocxref</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx&gt;8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_ocxref</td>
<td>'1'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_ocxref</td>
<td>'0'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Downcounting configuration**
  
  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the *Downcounting mode on page 1439.*

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then tim_ocxref is held at ‘1’. 0% PWM is not possible in this mode.

**PWM center-aligned mode**

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit...
(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to the Center-aligned mode (up/down counting) on page 1442.

Figure 371 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Figure 371. Center-aligned PWM waveforms (ARR=8)

Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit.
in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.

- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

### Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The **Figure 372** below presents the dithering principle applied to 4 consecutive PWM cycles.

![Figure 372. Dithering principle](MSv45752V1)

When the dithering mode is enabled, the register coding is changed as follows (see **Figure 373** for example):

- the 4 LSBs are coding for the enhanced resolution part (fractional part)
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value
Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 373. Data format and register coding in dithering mode

The minimum frequency is given by the following formula:

\[
\text{Resolution} = \frac{F_{\text{Tim}}}{F_{\text{pwm}}} \Rightarrow F_{\text{pwmMin}} = F_{\text{Tim}} \cdot F_{\text{Max Resolution}}
\]

Dithering mode disabled: \( F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65536} \)

Dithering mode enabled: \( F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65535 + \frac{15}{16}} \)

Note: The maximum TIMx ARR and TIMxCCRy values are limited to 0xFFFFE in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

As shown on the Figure 374 below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.
The duty cycle and/or period changes are spread over 16 consecutive periods, as described in the Figure 375 below.
The auto-reload and compare values increments are spread following specific patterns described in the Table 386 below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 386. CCR and ARR register change dithering pattern

<table>
<thead>
<tr>
<th>LSB value</th>
<th>PWM period</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0000</td>
<td>-</td>
</tr>
<tr>
<td>0001</td>
<td>+1</td>
</tr>
<tr>
<td>0010</td>
<td>+1</td>
</tr>
<tr>
<td>0011</td>
<td>+1</td>
</tr>
<tr>
<td>0100</td>
<td>+1</td>
</tr>
<tr>
<td>0101</td>
<td>+1</td>
</tr>
<tr>
<td>0110</td>
<td>+1</td>
</tr>
</tbody>
</table>
The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to '00'). In this case, the dithering pattern is applied over 8 consecutive PWM periods, considering the up and down counting phases as shown in the Figure 376 below.

![Figure 376. Dithering effect on duty cycle in center-aligned PWM mode](image)

Table 387 below shows how the dithering pattern is added in center-aligned PWM mode.

<table>
<thead>
<tr>
<th>LSB value</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0001</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0010</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0011</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
</tr>
<tr>
<td>0100</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0101</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
</tr>
<tr>
<td>0110</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
</tr>
<tr>
<td>0111</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 387. CCR register change dithering pattern in center-aligned PWM mode
### Table 387. CCR register change dithering pattern in center-aligned PWM mode (continued)

<table>
<thead>
<tr>
<th>LSB value</th>
<th>PWM period</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Up</td>
<td>Dn</td>
<td>Up</td>
<td>Dn</td>
<td>Up</td>
<td>Dn</td>
<td>Up</td>
<td>Dn</td>
<td>Up</td>
</tr>
<tr>
<td>0111</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
</tr>
<tr>
<td>1000</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
</tr>
<tr>
<td>1001</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
</tr>
<tr>
<td>1010</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
</tr>
<tr>
<td>1011</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>1100</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
</tr>
<tr>
<td>1101</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
</tr>
<tr>
<td>1110</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>1111</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
</tbody>
</table>

#### 38.3.14 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- `tim_oc1refc` (or `tim_oc2refc`) is controlled by TIMx_CCR1 and TIMx_CCR2
- `tim_oc3refc` (or `tim_oc4refc`) is controlled by TIMx_CCR3 and TIMx_CCR4

Asymmetric PWM mode can be selected independently on two channel (one `tim_ocx` output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

**Note:** The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an `tim_oc1refc` signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the `tim_oc2refc` signal on channel 2, or an `tim_oc2refc` signal resulting from asymmetric PWM mode 1.

*Figure 377* represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 2). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.
38.3.15 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 378 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1.
38.3.16 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The tim_oc5ref signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the tim_oc5ref is combined. The resulting signals, tim_ocxrefc, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, tim_oc1refc is controlled by TIMx_CCR1 and TIMx_CCR5
- If GC5C2 is set, tim_oc2refc is controlled by TIMx_CCR2 and TIMx_CCR5
- If GC5C3 is set, tim_oc3refc is controlled by TIMx_CCR3 and TIMx_CCR5

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].
The `tim_trgo2` waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to Section 38.3.31: ADC triggers for more details.

### 38.3.17 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1/TIM8) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...).

The polarity of the outputs (main output `tim_ocx` or complementary `tim_ocxn`) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals `tim_ocx` and `tim_ocxn` are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to Table 395: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature on page 1544 for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a
reference waveform `tim_ocxref`, it generates 2 outputs `tim_ocx` and `tim_ocxn`. If `tim_ocx` and `tim_ocxn` are active high:

- The `tim_ocx` output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The `tim_ocxn` output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (`tim_ocx` or `tim_ocxn`) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal `tim_ocxref`. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

**Figure 380. Complementary output with symmetrical dead-time insertion**

![Diagram showing complementary output with symmetrical dead-time insertion](MSv62332V1)

The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on Figure 381.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN=1.

It is possible to have the deadtime value updated on-the-fly during PWM operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIMx_DTR2 register. The preload value is loaded in the active register on the next update event.

**Note:** *If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.*
The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to Section 38.6.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8) for delay calculation.
Re-directing tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be re-directed to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

**Note:** When only tim_ocxn is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP=0 then tim_ocxref=tim_ocxn. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE=CCxNE=1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

38.3.18 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, ECC / parity errors,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register is used to enable /disable the outputs by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to Table 395: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature on page 1544 for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous
and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIMx_BKIN pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:
  - coming from a tim_brk_cmpx input (refer to Section 38.3.2: TIM1/TIM8 pins and internal signals for product specific implementation)
  - coming from a system break request (refer to Section 38.3.2: TIM1/TIM8 pins and internal signals for product specific implementation)

The sources for break2 (tim_brk2) are:

- External sources connected to one of the TIMx_BKIN2 pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources coming from a tim_brk2_cmpx input (refer to Section 38.3.2: TIM1/TIM8 pins and internal signals for product specific implementation)

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register.

All sources are ORed before entering the timer tim_brk or tim_brk2 inputs, as per Figure 384 below.
Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to...
their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 Tim_ker_ck clock cycles).

– If OSSi=0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

• The break status flag (SBIF, BIF and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.

• If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to ‘1’ again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSi value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

Note: The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to Section 38.6.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8). The LOCK bits can be written only once after an MCU reset.

Figure 385 shows an example of behavior of the outputs in response to a break.
The two break inputs have different behaviors on timer outputs:

- The `tim_brk` input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- `tim_brk2` can only disable (inactive state) the PWM outputs.
The `tim_brk` has a higher priority than `tim_brk2` input, as described in Table 388.

*Note:* `tim_brk2` must only be used with `OSSR = OSSI = 1`.

### Table 388. Behavior of timer outputs versus `tim_brk/tim_brk2` inputs

<table>
<thead>
<tr>
<th><code>tim_brk</code></th>
<th><code>tim_brk2</code></th>
<th>Timer outputs state</th>
<th>Typical use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>X</td>
<td>– Inactive then forced output state (after a deadtime) – Outputs disabled if <code>OSSI = 0</code> (control taken over by GPIO logic)</td>
<td>ON after deadtime insertion</td>
</tr>
<tr>
<td>Inactive</td>
<td>Active</td>
<td>Inactive</td>
<td>OFF</td>
</tr>
</tbody>
</table>

*Figure 386* gives an example of `tim_ocx` and `tim_ocxn` output behavior in case of active signals on `tim_brk` and `tim_brk2` inputs. In this case, both outputs have active high polarities (`CCxP = CCxNP = 0` in TIMx_CCER register).

*Figure 386. PWM output state following `tim_brk` and `tim_brk2` assertion (OSSI=1)*
38.3.19 Bidirectional break inputs

The TIM1/TIM8 are featuring bidirectional break I/Os, as represented on Figure 388.

This provides support for:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The tim_brk and tim_brk2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the tim_brk and tim_brk2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (for example CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to '0' to indicate to the external components that the timer is entered in break state. However, this is valid only if the break is enabled (BKE or B2KE = 1). When a software break event is generated with BKE or B2KE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the TIMx_BKin and TIMx_BKin2 I/Os.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BKDSRM (BK2DSRM) bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see Table 389).
Arming and re-arming break circuitry

The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

38.3.20 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocxref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM
cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode. tim_ocref_clr_int input can be selected between the tim_ocref_clr input and tim_etrf (tim_etrf_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register.

The tim_ocref_clr input can be selected among several inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown on the Figure 389 below. Refer to Section 38.3.2: TIM1/TIM8 pins and internal signals for a list of sources available in the product.

**Figure 389. tim_ocref_clr input selection multiplexer**

When tim_etrf is chosen, tim_etrf_in must be configured as follows:

1. The External Trigger Prescaler must be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to ‘00’.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to ‘0’.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to application needs (as per polarity of the source connected to the trigger and eventual need to remove noise using the filter).

*Figure 390* shows the behavior of the tim_ocxref signal when the tim_etrf Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.
Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then tim_ocxref is enabled again at the next counter overflow.

38.3.21 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The Figure 391 describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.
38.3.22 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: $\text{CNT} < \text{CCRx} \leq \text{ARR}$ (in particular, $0 < \text{CCRx}$)
- In downcounting: $\text{CNT} > \text{CCRx}$
For example one may want to generate a positive pulse on `tim_oc1` with a length of `tPULSE` and after a delay of `tDELAY` as soon as a positive edge is detected on the `tim_ti2` input pin.

Let's use `tim_ti2fp2` as trigger 1:
- Map `tim_ti2fp2` to `tim_ti2` by writing `CC2S='01'` in the `TIMx_CCMR1` register.
- `tim_ti2fp2` must detect a rising edge, write `CC2P='0'` and `CC2NP='0'` in the `TIMx_CCER` register.
- Configure `tim_ti2fp2` as trigger for the slave mode controller (`tim_trgi`) by writing `TS=00110` in the `TIMx_SMCR` register.
- `tim_ti2fp2` is used to start the counter by writing `SMS to '110'` in the `TIMx_SMCR` register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).
- The `tDELAY` is defined by the value written in the `TIMx_CCR1` register.
- The `tPULSE` is defined by the difference between the auto-reload value and the compare value (`TIMx_ARR - TIMx_CCR1`).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing `OC1M=111` in the `TIMx_CCMR1` register. Optionally the preload registers can be enabled by writing `OC1PE='1'` in the `TIMx_CCMR1` register and ARPE in the `TIMx_CR1` register. In this case one has to write the compare value in the `TIMx_CCR1` register, the auto-reload value in the `TIMx_ARR` register, generate an update by setting the UG bit and wait for external trigger event on `tim_ti2`. `CC1P` is written to '0' in this example.

In our example, the DIR and CMS bits in the `TIMx_CR1` register must be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the `TIMx_CR1` register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the `TIMx_CR1` register is set to '0', so the Repetitive Mode is selected.
Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay $t_{\text{DELAY min}}$ we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 38.3.23 Retriggerable One-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one-pulse mode described in Section 38.3.22:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

**Note:** The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.
38.3.24 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulsewidth generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on the Figure 394 below.

Figure 394. Pulse generator circuitry

The Figure 395 below shows how the pulse is generated for edge-aligned and encoder operating modes.
This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bit fields in TIMx_CCMR2 register.

The pulselength is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

\[ t_{PW} = PW[7:0] \times t_{PWG} \]

where \( t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim\_ker\_ck} \).

This gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

**Note:** If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the 1st arriving trigger is extended (because of the re-trigger), while the pulse width of the last arriving trigger is correct (as shown on the Figure 396 below).
38.3.25 Encoder interface mode

**Quadrature encoder**

To select Encoder Interface mode write SMS='0001' in the TIMx_SMCR register if the counter is counting on tim_t1 edges only, SMS='0010' if it is counting on tim_t2 edges only and SMS='0011' if it is counting on both tim_t1 and tim_t2 edges.

Select the tim_t1 and tim_t2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_t1 and tim_t2 are used to interface to a quadrature encoder. Refer to Table 390. The counter is clocked by each valid transition on tim_t1fp1 or tim_t2fp2 (tim_t1 and tim_t2 after input filter and polarity selection, tim_t1fp1=tim_t1 if not filtered and not inverted, tim_t2fp2=tim_t2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to ‘1’). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_t1 or tim_t2), whatever the counter is counting on tim_t1 only, tim_t2 only or both tim_t1 and tim_t2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx.ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder’s position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_t1 and tim_t2 do not switch at the same time.
A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder’s differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

The Figure 397 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S='01' (TIMx_CCMR2 register, tim_ti1fp2 mapped on tim_ti2).
- CC1P='0' and CC1NP='0' (TIMx_CCER register, tim_ti1fp1 non-inverted, tim_ti1fp1=tim_ti1).
- CC2P='0' and CC2NP='0' (TIMx_CCER register, tim_ti1fp2 non-inverted, tim_ti1fp2=tim_ti2).
- SMS='0011' (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (TIMx_CR1 register, Counter enabled).

<table>
<thead>
<tr>
<th>Table 390. Counting direction versus encoder signals (CC1P = CC2P = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active edge</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>Counting on tim_ti1 only x1 mode</td>
</tr>
<tr>
<td>Counting on tim_ti2 only x1 mode</td>
</tr>
<tr>
<td>Counting on tim_ti1 only x2 mode</td>
</tr>
<tr>
<td>Counting on tim_ti2 only x2 mode</td>
</tr>
<tr>
<td>Counting on tim_ti1 and tim_ti2 x4 mode</td>
</tr>
</tbody>
</table>
Figure 397. Example of counter operation in encoder interface mode.

Figure 398 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P='1').

Figure 398. Example of encoder interface mode with tim_ti1fp1 polarity inverted.

The Figure 399 below shows the timer counter value during a speed reversal, for various counting modes.
The timer, when configured in Encoder Interface mode provides information on the sensor’s current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register’s bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter’s most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support other types of encoders.

In the “clock plus direction” mode shown on Figure 400, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity
The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

**Figure 400. Direction plus clock encoder mode**

<table>
<thead>
<tr>
<th>Directional Clock encoder mode</th>
</tr>
</thead>
</table>

In the “directional clock” mode on Figure 401, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- **1100:** x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to Figure 401 below) and CCxP = 1 corresponds to low-level idle state (refer to Figure 402 below).

- **1101:** x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to Figure 401 below), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to Figure 402 below).

**Figure 401. Directional clock encoder mode (CC1P = CC2P = 0)**
Figure 402. Directional clock encoder mode (CC1P = CC2P = 1)

The Table 391 here-below details how the directional clock mode operates, for any input transition.

Table 391. Counting direction versus encoder signals and polarity settings

<table>
<thead>
<tr>
<th>Directional clock mode</th>
<th>SMS[3:0]</th>
<th>Level on opposite signal (tim_ti1fp1 for tim_ti1, tim_ti2fp2 for tim_ti2)</th>
<th>tim_ti1fp1 signal</th>
<th>tim_ti2fp2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>tim_ti1fp1 for tim_ti1, tim_ti2fp2 for tim_ti2</td>
<td>Rising</td>
<td>Falling</td>
</tr>
<tr>
<td>x2 mode</td>
<td>1100</td>
<td>High</td>
<td>Down</td>
<td>Down</td>
</tr>
<tr>
<td>CCxP=0</td>
<td></td>
<td>Low</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>x2 mode</td>
<td>1100</td>
<td>High</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>CCxP=1</td>
<td></td>
<td>Low</td>
<td>Down</td>
<td>Up</td>
</tr>
<tr>
<td>x1 mode</td>
<td>1101</td>
<td>High</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>CCxP=0</td>
<td></td>
<td>Low</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>x1 mode</td>
<td>1101</td>
<td>High</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>CCxP=1</td>
<td></td>
<td>Low</td>
<td>Down</td>
<td>No count</td>
</tr>
</tbody>
</table>

**Index Input**

The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The Index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMX_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.
Commercially available encoders are proposed with several options for index pulse conditioning, as per the Figure 403 below:

- gated with A and B: the pulsewidth is $1/4$ of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulsewidth is $1/2$ of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulsewidth is up to one channel period, without any alignment to the edges

**Figure 403. Index gating options**

The circuitry tolerates jitter on index signal, whatever the gating mode, as shown on Figure 404 below.

In ungated mode, the signal must be strictly below 2 encoder periods. If the pulsewidth is greater or equal to 2 encoder period, the counter is reset multiple times.

**Figure 404. Jittered Index signals**

The timer supports the 3 gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (i.e. channel A and channel B
state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The Index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. The Figure 405 below shows at which position is the index generated, for a simplistic example (an encoder providing 4 edges par mechanical rotation).

**Figure 405. Index generation for IPOS[1:0] = 11**

The Figure 406 below presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is ‘11’ (ChA=1, ChB=1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the ‘11’ state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.
Figure 406. Counter reading with index gated on channel A (IPOS[1:0] = 11)

The Figure 407. below presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 407. Counter reading with index ungated (IPOS[1:0] = 00)

The Figure 408. below shows how the ‘gated on A & B’ mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.
The *Figure 409* and *Figure 410* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.
Figure 409. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)
The Figure 411 below shows how the index is managed in x1 and x2 modes.
**Figure 411. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)**

![Diagram](image)

**Directional index sensitivity**

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

The *Figure 412* below shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

**Figure 412. Directional index sensitivity**

![Diagram](image)
**Special first index event management**

The FIDX bit in the TIMx_ECR register allows the Index to be taken only once, as shown on the *Figure 413* below. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be re-armed by writing the FIDX bit to 0 and setting it again to 1.

*Figure 413. Counter reset as function of FIDX bit setting*

---

**Index blanking**

The Index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on the *Figure 414* below.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input

*Figure 414. Index blanking*
Index management in non-quadrature mode

The Figure 415 and Figure 416 below detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:
- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

**Figure 415. Index behavior in clock + direction mode, IPOS[0] = 1**

**Figure 416. Index behavior in directional clock mode, IPOS[0] = 1**

Encoder error management

For encoder configurations where 2 quadrature signals are available, it is possible to detect transition errors. The reading on the 2 inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on the Figure 417. below. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR status.
register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

**Figure 417. State diagram for quadrature encoded signals**

For encoder having an Index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The Index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an Index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is to cope with narrow index pulses in gated A and B mode, as shown on Figure 418 below.

**Figure 418. Up-counting encoder error detection**
In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on Figure 419 below, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 419. Down-counting encode error detection

An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder Interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the Index event sets the IDXF interrupt flag in the TIMx_SR status register. An Index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the Update interrupt rate, by switching from x4 to x2 to x1 mode, as show on the Figure 420 below.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- **SMSPS = 0**: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting.
- **SMSPS = 1**: the transfer is triggered by the Index event.

![Figure 420. Encoder mode change with preload transferred on update (SMSPS = 0)](image)

**Encoder clock output**

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register. It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

38.3.26 Direction bit output

It is possible to output a direction signal out of the timer, on the tim_oc3n and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bit field to 1011 in the TIMx_CCMR2 register.
This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

### 38.3.27 UIF bit remapping

The UIFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMx_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the UIF and UIFCPY flags assertion.

### 38.3.28 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3. The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per Figure 421 below.

**Figure 421. Measuring time interval between edges on 3 signals**

![Figure 421](MSv62359V2)

### 38.3.29 Interfacing with Hall sensors

This is done using the advanced-control timers to generate PWM signals to drive the motor and another timer TIMx referred to as “interfacing timer” in Figure 422. The “interfacing timer” captures the 3 timer input pins (tim_ti1, tim_ti2 and tim_ti3) connected through a XOR to the tim_ti1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is tim_ti1f_ed. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is tim_trc (See Figure 364: Capture/compare channel (example: channel 1...
The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (by triggering a COM event). The advanced-control timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer through the tim_trgo output.

Example: one wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the tim_ti1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’,
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the tim_ti1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (tim_trc selected): write the CC1S bits in the TIMx_CCMR1 register to ‘01’. The digital filter can also be programmed if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to ‘111’ and the CC2S bits to ‘00’ in the TIMx_CCMR1 register,
- Select tim_oc2ref as trigger output on tim_trgo: write the MMS bits in the TIMx_CR2 register to ‘101’.

In the advanced-control timer, the right tim_itrx input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of tim_oc2ref).

The Figure 422 describes this example.
38.3.30 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to Section 39.4.23: Timer synchronization for details. They can be synchronized in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

- Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
- Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

**Figure 423. Control circuit in reset mode**

Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when tim_ti1 input is low:

- Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN=0, whatever is the trigger input level).
The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

**Slave mode: Trigger mode**

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

- Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.
Slave mode: Combined reset + trigger mode
In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.
This mode is used for One-pulse mode.

Slave mode: Combined gated + reset mode
The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode: external clock mode 2 + trigger mode
The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_tr1 occurs:
1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on tim_etr_in and ECE=1 to enable the external clock mode 2.

2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01 in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edge only).

3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.

   A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

   The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

**Figure 426. Control circuit in external clock mode 2 + trigger mode**

### Note:
The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.
38.3.31 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the tim_trgo2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in Figure 379 on page 1471.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

38.3.32 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR register define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see Section 38.6.29: TIMx DMA control register (TIMx_DCR)(x = 1, 8) for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:
1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (see note below).
   - Circular mode disabled.

2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:
   DBL = 3 transfers, DBA = 0xE and DBSS = 1.

3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).

4. Enable TIMx.

5. Enable the DMA channel.

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

38.3.33 TIM1/TIM8 DMA requests

The TIM1/TIM8 can generate a DMA request, as shown in the table below.

<table>
<thead>
<tr>
<th>DMA request signal</th>
<th>DMA request</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_upd_dma</td>
<td>Update</td>
<td>UDE</td>
</tr>
<tr>
<td>tim_cc1_dma</td>
<td>Capture/compare 1</td>
<td>CC1DE</td>
</tr>
<tr>
<td>tim_cc2_dma</td>
<td>Capture/compare 2</td>
<td>CC2DE</td>
</tr>
<tr>
<td>tim_cc3_dma</td>
<td>Capture/compare 3</td>
<td>CC3DE</td>
</tr>
<tr>
<td>tim_cc4_dma</td>
<td>Capture/compare 4</td>
<td>CC4DE</td>
</tr>
<tr>
<td>tim_com_dma</td>
<td>Commutation (COM)</td>
<td>COMDE</td>
</tr>
<tr>
<td>tim_trg_dma</td>
<td>Trigger</td>
<td>TDE</td>
</tr>
</tbody>
</table>

38.3.34 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop, depending on DBG_TIMx_STOP configuration bit in DBG module.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.
For more details, refer to section Debug support (DBG).

### 38.4 TIM1/TIM8 low-power modes

**Table 393. Effect of low-power modes on TIM1/TIM8**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The timer operation is stopped and the register content is kept. No interrupt can be generated.</td>
</tr>
<tr>
<td>Standby</td>
<td>The timer is powered-down and must be reinitialized after exiting the Standby mode.</td>
</tr>
</tbody>
</table>

### 38.5 TIM1/TIM8 interrupts

The TIM1/TIM8 can generate multiple interrupts, as shown in **Table 394**.

**Table 394. Interrupt requests**

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop and Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_UP</td>
<td>Update</td>
<td>UIF</td>
<td>UIE</td>
<td>write 0 in UIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_CC</td>
<td>Capture/compare 1</td>
<td>CC1IF</td>
<td>CC1IE</td>
<td>write 0 in CC1IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 2</td>
<td>CC2IF</td>
<td>CC2IE</td>
<td>write 0 in CC2IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 3</td>
<td>CC3IF</td>
<td>CC3IE</td>
<td>write 0 in CC3IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 4</td>
<td>CC4IF</td>
<td>CC4IE</td>
<td>write 0 in CC4IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_TRG_COM</td>
<td>Commutation (COM)</td>
<td>COMIF</td>
<td>COMIE</td>
<td>write 0 in COMIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Trigger</td>
<td>TIF</td>
<td>TIE</td>
<td>write 0 in TIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_DIR_IDX</td>
<td>Index</td>
<td>IDXF</td>
<td>IDXIE</td>
<td>write 0 in IDXF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Direction</td>
<td>DIRF</td>
<td>DIRIE</td>
<td>write 0 in DIRF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_BRK</td>
<td>Break</td>
<td>BIF</td>
<td>BIE</td>
<td>write 0 in BIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Break2</td>
<td>B2IF</td>
<td></td>
<td>write 0 in B2IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>System Break</td>
<td>SBIF</td>
<td></td>
<td>write 0 in SBIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_IERR</td>
<td>Index Error</td>
<td>IERRF</td>
<td>IERRIE</td>
<td>write 0 in IERRF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_TER</td>
<td>Transition Error</td>
<td>TERRF</td>
<td>TERRIE</td>
<td>write 0 in TERRF</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
### 38.6 TIM1/TIM8 registers

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

#### 38.6.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8)

Address offset: 0x000

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:13</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 12</td>
<td>DITHEN: Dithering enable</td>
</tr>
<tr>
<td></td>
<td>0: Dithering disabled</td>
</tr>
<tr>
<td></td>
<td>1: Dithering enabled</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The DITHEN bit can only be modified when CEN bit is reset.</td>
</tr>
<tr>
<td>Bit 11</td>
<td>UIFREMAP: UIF status bit remapping</td>
</tr>
<tr>
<td></td>
<td>0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.</td>
</tr>
<tr>
<td></td>
<td>1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.</td>
</tr>
<tr>
<td>Bit 10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 9:8</td>
<td>CKD[1:0]: Clock division</td>
</tr>
<tr>
<td></td>
<td>This bit-field indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (tDTS) used by the dead-time generators and the digital filters (stim_etr_in, tim_tix).</td>
</tr>
<tr>
<td></td>
<td>00: tDTS=tim_ker_ck</td>
</tr>
<tr>
<td></td>
<td>01: tDTS=2*tim_ker_ck</td>
</tr>
<tr>
<td></td>
<td>10: tDTS=4*tim_ker_ck</td>
</tr>
<tr>
<td></td>
<td>11: Reserved, do not program this value</td>
</tr>
<tr>
<td>Bit 7</td>
<td>ARPE: Auto-reload preload enable</td>
</tr>
<tr>
<td></td>
<td>0: TIMx_ARR register is not buffered</td>
</tr>
<tr>
<td></td>
<td>1: TIMx_ARR register is buffered</td>
</tr>
<tr>
<td>Bit 6:5</td>
<td>CMS[1:0]: Center-aligned mode selection</td>
</tr>
<tr>
<td></td>
<td>00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).</td>
</tr>
<tr>
<td></td>
<td>01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.</td>
</tr>
<tr>
<td></td>
<td>10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.</td>
</tr>
<tr>
<td></td>
<td>11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)</td>
</tr>
</tbody>
</table>
Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter

*Note:* This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled.
These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value
(ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is
set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled

*Note:* External clock, gated mode and encoder mode can work only if the CEN bit has been
previously set by software. However trigger mode can set the CEN bit automatically by
hardware.

### 38.6.2 TIMx control register 2 (TIMx_CR2)(x = 1, 8)

Address offset: 0x004
Reset value: 0x0000 0000
Bits 31:26  Reserved, must be kept at reset value.
Bit 24  Reserved, must be kept at reset value.

Bits 23:20  **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (tim_trgo2) to be selected. The combination is as follows:

0000:  **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on tim_trgo2 is delayed compared to the actual reset.
0001:  **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).
0010:  **Update** - the update event is selected as trigger output (tim_trgo2). For instance, a master timer can then be used as a prescaler for a slave timer.
0011:  **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (tim_trgo2).
0100:  **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo2)
0101:  **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo2)
0110:  **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo2)
0111:  **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo2)
1000:  **Compare** - tim_oc5refc signal is used as trigger output (tim_trgo2)
1001:  **Compare** - tim_oc6refc signal is used as trigger output (tim_trgo2)
1010:  **Compare Pulse** - tim_oc4refc rising or falling edges generate pulses on tim_trgo2
1011:  **Compare pulse** - tim_oc6refc rising or falling edges generate pulses on tim_trgo2
1100:  **Compare pulse** - tim_oc4refc or tim_oc6refc rising edges generate pulses on tim_trgo2
1101:  **Compare pulse** - tim_oc4refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2
1110:  **Compare pulse** - tim_oc5refc or tim_oc6refc rising edges generate pulses on tim_trgo2
1111:  **Compare pulse** - tim_oc5refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19  Reserved, must be kept at reset value.

Bit 18  **OIS6**: Output idle state 6 (tim_oc6 output)
Refer to OIS1 bit

Bit 17  Reserved, must be kept at reset value.

Bit 16  **OIS5**: Output idle state 5 (tim_oc5 output)
Refer to OIS1 bit

Bit 15  **OIS4N**: Output idle state 4 (tim_oc4n output)
Refer to OIS1N bit

Bit 14  **OIS4**: Output idle state 4 (tim_oc4 output)
Refer to OIS1 bit
Bit 13  **OIS3N**: Output idle state 3 (tim_oc3n output)
        Refer to OIS1N bit

Bit 12  **OIS3**: Output idle state 3 (tim_oc3n output)
        Refer to OIS1 bit

Bit 11  **OIS2N**: Output idle state 2 (tim_oc2n output)
        Refer to OIS1N bit

Bit 10  **OIS2**: Output idle state 2 (tim_oc2 output)
        Refer to OIS1 bit

Bit 9   **OIS1N**: Output idle state 1 (tim_oc1n output)
        0: tim_oc1n=0 after a dead-time when MOE=0
        1: tim_oc1n=1 after a dead-time when MOE=0
        **Note**: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8   **OIS1**: Output idle state 1 (tim_oc1 output)
        0: tim_oc1=0 (after a dead-time) when MOE=0
        1: tim_oc1=1 (after a dead-time) when MOE=0
        **Note**: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7   **TI1S**: tim_ti1 selection
        0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
        1: tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input
Bits 25, 6:4 **MMS[3:0]**: Master mode selection

These bits select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

- **0000**: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

- **0001**: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

- **0010**: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

- **0011**: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

- **0100**: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

- **0101**: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

- **0110**: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

- **0111**: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

- **1000**: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Other codes reserved

**Note:** The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

- 0: CCx DMA request sent when CCx event occurs
- 1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

- 0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only
- 1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi

**Note:** This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

- 0: CCxE, CCxNE and OCxM bits are not preloaded
- 1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

**Note:** This bit acts only on channels that have a complementary output.
### 38.6.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)

Address offset: 0x008  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:26</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 25</td>
<td><strong>SMSPS</strong>: SMS preload source</td>
</tr>
<tr>
<td></td>
<td>This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active</td>
</tr>
<tr>
<td></td>
<td>0: The transfer is triggered by the Timer’s Update event</td>
</tr>
<tr>
<td></td>
<td>1: The transfer is triggered by the Index event</td>
</tr>
<tr>
<td>Bit 24</td>
<td><strong>SMSPE</strong>: SMS preload enable</td>
</tr>
<tr>
<td></td>
<td>This bit selects whether the SMS[3:0] bitfield is preloaded</td>
</tr>
<tr>
<td></td>
<td>0: SMS[3:0] bitfield is not preloaded</td>
</tr>
<tr>
<td></td>
<td>1: SMS[3:0] preload is enabled</td>
</tr>
<tr>
<td>Bit 23:22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 21:20</td>
<td><strong>TS[4:3]</strong>: Trigger selection - bit 4:3</td>
</tr>
<tr>
<td></td>
<td>Refer to TS[2:0] description - bits 6:4</td>
</tr>
<tr>
<td>Bit 19:17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 15</td>
<td><strong>ETP</strong>: External trigger polarity</td>
</tr>
<tr>
<td></td>
<td>This bit selects whether tim_etr_in or tim_etr_in is used for trigger operations</td>
</tr>
<tr>
<td></td>
<td>0: tim_etr_in is non-inverted, active at high level or rising edge.</td>
</tr>
<tr>
<td></td>
<td>1: tim_etr_in is inverted, active at low level or falling edge.</td>
</tr>
<tr>
<td>Bit 14</td>
<td><strong>ECE</strong>: External clock enable</td>
</tr>
<tr>
<td></td>
<td>This bit enables External clock mode 2.</td>
</tr>
<tr>
<td></td>
<td>0: External clock mode 2 disabled</td>
</tr>
<tr>
<td></td>
<td>1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.</td>
</tr>
</tbody>
</table>

**Note:** Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etr (SMS=111 and TS=00111).  
It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etr in this case (TS bits must not be 0011).  
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etr.
Bits 13:12 ETPS[1:0]: External trigger prescaler
   External trigger signal tim_etrp frequency must be at most 1/4 of TIMxCLK frequency. A
   prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast
   external clocks on tim_etrl_in.
   00: Prescaler OFF
   01: tim_etrl_in frequency divided by 2
   10: tim_etrl_in frequency divided by 4
   11: tim_etrl_in frequency divided by 8

Bits 11:8 ETF[3:0]: External trigger filter
   This bit-field then defines the frequency used to sample tim_etrp signal and the length of the
   digital filter applied to tim_etrp. The digital filter is made of an event counter in which N
   consecutive events are needed to validate a transition on the output:
   0000: No filter, sampling is done at fDTS
   0001: fSAMPLING=ftim_ker_ck, N=2
   0010: fSAMPLING=ftim_ker_ck, N=4
   0011: fSAMPLING=ftim_ker_ck, N=8
   0100: fSAMPLING=fDTS/2, N=6
   0101: fSAMPLING=fDTS/2, N=8
   0110: fSAMPLING=fDTS/4, N=6
   0111: fSAMPLING=fDTS/4, N=8
   1000: fSAMPLING=fDTS/8, N=6
   1001: fSAMPLING=fDTS/8, N=8
   1010: fSAMPLING=fDTS/16, N=5
   1011: fSAMPLING=fDTS/16, N=6
   1100: fSAMPLING=fDTS/32, N=5
   1101: fSAMPLING=fDTS/32, N=6
   1110: fSAMPLING=fDTS/32, N=8
   1111: fSAMPLING=fDTS/32, N=8

Bit 7 MSM: Master/slave mode
   0: No action
   1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect
      synchronization between the current timer and its slaves (through tim_trgo). It is useful if
      we want to synchronize several timers on a single external event.
Bits 6:4 **TS[2:0]: Trigger selection**

This bitfield is combined with TS[4:3] bits. This bit-field selects the trigger input to be used to synchronize the counter.

- **00000**: Internal Trigger 0 (tim_itr0)
- **00001**: Internal Trigger 1 (tim_itr1)
- **00010**: Internal Trigger 2 (tim_itr2)
- **00011**: Internal Trigger 3 (tim_itr3)
- **00100**: tim_ti1 Edge Detector (tim_ti1f_ed)
- **00101**: Filtered Timer Input 1 (tim_ti1fp1)
- **00110**: Filtered Timer Input 2 (tim_ti2fp2)
- **00111**: External Trigger input (tim_etrf)
- **01000**: Internal Trigger 4 (tim_itr4)
- **01001**: Internal Trigger 5 (tim_itr5)
- **01010**: Internal Trigger 6 (tim_itr6)
- **01011**: Internal Trigger 7 (tim_itr7)
- **01100**: Internal Trigger 8 (tim_itr8)
- **01101**: Internal Trigger 9 (tim_itr9)
- **01110**: Internal Trigger 10 (tim_itr10)
- **01111**: Internal trigger 11 (tim_itr11)
- **10000**: Internal trigger 12 (tim_itr12)
- **10001**: Internal trigger 13 (tim_itr13)
- **10010**: Internal trigger 14 (tim_itr14)
- **10011**: Internal trigger 15 (tim_itr15)
- **Others**: Reserved

See [Table 381: Internal trigger connection](#) for more details on tim_itrx meaning for each Timer.

**Note:** These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS**: OREF clear selection

This bit is used to select the OCREF clear source.

- **0**: tim_ocref_clr_int is connected to the tim_ocref_clr input
- **1**: tim_ocref_clr_int is connected to tim_etrf
Bits 16, 2:0  **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.

0001: Quadrature encoder mode 1, x2 mode - Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P.

1100: Encoder mode: Directional Clock, x2 mode.

1101: Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

**Note**: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS=00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

**Note**: The clock of the slave peripherals (timer, ADC, …) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.
### 38.6.4 TIMx DMA/interrupt enable register (TIMx_DIER) (x = 1, 8)

Address offset: 0x00C  
Reset value: 0x0000 0000

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

Bits 31:24 Reserved, must be kept at reset value.

- **Bit 23 TERRIE**: Transition error interrupt enable  
  0: Transition error interrupt disabled  
  1: Transition error interrupt enabled

- **Bit 22 IERRIE**: Index error interrupt enable  
  0: Index error interrupt disabled  
  1: Index error interrupt enabled

- **Bit 21 DIRIE**: Direction change interrupt enable  
  0: Direction Change interrupt disabled  
  1: Direction Change interrupt enabled

- **Bit 20 IDXIE**: Index interrupt enable  
  0: Index interrupt disabled  
  1: Index Change interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

- **Bit 14 TDE**: Trigger DMA request enable  
  0: Trigger DMA request disabled  
  1: Trigger DMA request enabled

- **Bit 13 COMDE**: COM DMA request enable  
  0: COM DMA request disabled  
  1: COM DMA request enabled

- **Bit 12 CC4DE**: Capture/compare 4 DMA request enable  
  0: CC4 DMA request disabled  
  1: CC4 DMA request enabled

- **Bit 11 CC3DE**: Capture/compare 3 DMA request enable  
  0: CC3 DMA request disabled  
  1: CC3 DMA request enabled

- **Bit 10 CC2DE**: Capture/compare 2 DMA request enable  
  0: CC2 DMA request disabled  
  1: CC2 DMA request enabled

- **Bit 9 CC1DE**: Capture/compare 1 DMA request enable  
  0: CC1 DMA request disabled  
  1: CC1 DMA request enabled
Bit 8 **UDE**: Update DMA request enable  
0: Update DMA request disabled  
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable  
0: Break interrupt disabled  
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable  
0: Trigger interrupt disabled  
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable  
0: COM interrupt disabled  
1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable  
0: CC4 interrupt disabled  
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable  
0: CC3 interrupt disabled  
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable  
0: CC2 interrupt disabled  
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable  
0: CC1 interrupt disabled  
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable  
0: Update interrupt disabled  
1: Update interrupt enabled

### 38.6.5 TIMx status register (TIMx_SR)(x = 1, 8)

Address offset: 0x010  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TERRF</td>
<td>IERRF</td>
<td>DIRF</td>
<td>IDXF</td>
<td></td>
<td></td>
<td></td>
<td>CC6IF</td>
<td>CC5IF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td></td>
<td>rc_w0</td>
<td>rc_w0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 15</td>
<td>Bit 14</td>
<td>Bit 13</td>
<td>Bit 12</td>
<td>Bit 11</td>
<td>Bit 10</td>
<td>Bit 9</td>
<td>Bit 8</td>
<td>Bit 7</td>
<td>Bit 6</td>
<td>Bit 5</td>
<td>Bit 4</td>
<td>Bit 3</td>
<td>Bit 2</td>
<td>Bit 1</td>
<td>Bit 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
</tr>
</tbody>
</table>
Bits 31:24  Reserved, must be kept at reset value.

Bit 23  **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to '0'.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22  **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to '0'.
0: No index error has been detected.
1: An index error has been detected

Bit 21  **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to '0'.
0: No direction change
1: Direction change

Bit 20  **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to '0'.
0: No index event occurred.
1: An index event has occurred

Bits 19:18  Reserved, must be kept at reset value.

Bit 17  **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description

*Note: Channel 6 can only be configured as output.*

Bit 16  **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description

*Note: Channel 5 can only be configured as output.*

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

Bit 13  **SBIF**: System break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 12  **CC4OF**: Capture/compare 4 overcapture flag
Refer to CC1OF description

Bit 11  **CC3OF**: Capture/compare 3 overcapture flag
Refer to CC1OF description

Bit 10  **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description
Bit 9  **CC1OF**: Capture/compare 1 overcapture flag
   - This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
   - 0: No overcapture has been detected.
   - 1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set.

Bit 8  **B2IF**: Break 2 interrupt flag
   - This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.
   - 0: No break event occurred.
   - 1: An active level has been detected on the break 2 input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 7  **BIF**: Break interrupt flag
   - This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
   - 0: No break event occurred.
   - 1: An active level has been detected on the break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 6  **TIF**: Trigger interrupt flag
   - This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
   - 0: No trigger event occurred.
   - 1: Trigger interrupt pending.

Bit 5  **COMIF**: COM interrupt flag
   - This flag is set by hardware on COM event (when capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.
   - 0: No COM event occurred.
   - 1: COM interrupt pending.

Bit 4  **CC4IF**: Capture/compare 4 interrupt flag
   - Refer to CC1IF description

Bit 3  **CC3IF**: Capture/compare 3 interrupt flag
   - Refer to CC1IF description
38.6.6 TIMx event generation register (TIMx_EGR)(x = 1, 8)

Address offset: 0x014
Reset value: 0x0000

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

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

Bit 8 **B2G**: Break 2 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.
Bit 6  **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5  **COMG**: Capture/compare control update generation
This bit can be set by software, it is automatically cleared by hardware
0: No action
1: CCxE, CCxNE and OCxM bits update (providing CCPC bit is set)
*Note*: This bit acts only on channels having a complementary output.

Bit 4  **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3  **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2  **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1  **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
*If channel CC1 is configured as output:*
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
*If channel CC1 is configured as input:*
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0  **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

### 38.6.7  TIMx capture/compare mode register 1 (TIMx_CCMR1)  
(*x = 1, 8*)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).
### Input capture mode:

Bits 31:16  **Reserved, must be kept at reset value.**

Bits 15:12  **IC2F[3:0]:** Input capture 2 filter

Bits 11:10  **IC2PSC[1:0]:** Input capture 2 prescaler

Bits 9:8   **CC2S[1:0]:** Capture/compare 2 selection
- This bit-field defines the direction of the channel (input/output) as well as the used input.
  - 00: CC2 channel is configured as output
  - 01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
  - 10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
  - 11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

**Note:** CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).
Advanced-control timers (TIM1/TIM8) RM0481

Bits 7:4 IC1F[3:0]: Input capture 1 filter
This bit-field defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:
- 0000: No filter, sampling is done at fDTS
- 0001: fSAMPLING=fTim_ker_ck, N=2
- 0010: fSAMPLING=fTim_ker_ck, N=4
- 0011: fSAMPLING=fTim_ker_ck, N=8
- 0100: fSAMPLING=fDTS/2, N=6
- 0101: fSAMPLING=fDTS/2, N=8
- 0110: fSAMPLING=fDTS/4, N=6
- 0111: fSAMPLING=fDTS/4, N=8
- 1000: fSAMPLING=fDTS/8, N=6
- 1001: fSAMPLING=fDTS/8, N=8
- 1010: fSAMPLING=fDTS/16, N=5
- 1011: fSAMPLING=fDTS/16, N=6
- 1100: fSAMPLING=fDTS/16, N=8
- 1101: fSAMPLING=fDTS/32, N=5
- 1110: fSAMPLING=fDTS/32, N=6
- 1111: fSAMPLING=fDTS/32, N=8

Bits 3:2 IC1PSC[1:0]: Input capture 1 prescaler
This bit-field defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
- 00: no prescaler, capture is done each time an edge is detected on the capture input
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0 CC1S[1:0]: Capture/compare 1 Selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
- 10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
- 11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

38.6.8 TIMx capture/compare mode register 1 [alternate]
(TIMx_CCMR1)(x = 1, 8)
Address offset: 0x018
Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).
Output compare mode:

Bits 31:25  Reserved, must be kept at reset value.

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

Bit 15   \textbf{OC2CE}: Output compare 2 clear enable

Bits 24, 14:12  \textbf{OC2M[3:0]}: Output compare 2 mode

Bit 11   \textbf{OC2PE}: Output compare 2 preload enable

Bit 10   \textbf{OC2FE}: Output compare 2 fast enable

Bits 9:8   \textbf{CC2S[1:0]}: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

- 00: CC2 channel is configured as output
- 01: CC2 channel is configured as input, tim\_ic2 is mapped on tim\_ti2
- 10: CC2 channel is configured as input, tim\_ic2 is mapped on tim\_ti1
- 11: CC2 channel is configured as input, tim\_ic2 is mapped on tim\_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx\_SMCR register)

\textbf{Note}: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx\_CCER).

Bit 7   \textbf{OC1CE}: Output compare 1 clear enable

0: tim\_oc1ref is not affected by the tim\_ocref\_clr\_int signal
1: tim\_oc1ref is cleared as soon as a High level is detected on tim\_ocref\_clr\_int signal (tim\_ocref\_clr input or tim\_etrf input)
Bits 16, 6:4 **OC1M[3:0]: Output compare 1 mode**

These bits define the behavior of the output reference signal `tim_oc1ref` from which `tim_oc1` and `tim_oc1n` are derived. `tim_oc1ref` is active high whereas `tim_oc1` and `tim_oc1n` active level depends on `CC1P` and `CC1NP` bits.

0000: Frozen - The comparison between the output compare register `TIMx_CCR1` and the counter `TIMx_CNT` has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. `tim_oc1ref` signal is forced high when the counter `TIMx_CNT` matches the capture/compare register 1 (`TIMx_CCR1`).

0010: Set channel 1 to inactive level on match. `tim_oc1ref` signal is forced low when the counter `TIMx_CNT` matches the capture/compare register 1 (`TIMx_CCR1`).

0011: Toggle - `tim_oc1ref` toggles when `TIMx_CNT=TIMx_CCR1`.

0100: Force inactive level - `tim_oc1ref` is forced low.

0101: Force active level - `tim_oc1ref` is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as `TIMx_CNT<TIMx_CCR1` else inactive. In downcounting, channel 1 is inactive (`tim_oc1ref='0'`) as long as `TIMx_CNT>TIMx_CCR1` else active (`tim_oc1ref='1'`).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as `TIMx_CNT<TIMx_CCR1` else active. In downcounting, channel 1 is active as long as `TIMx_CNT>TIMx_CCR1` else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on `tim_trgi` signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on `tim_trgi` signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on `tim_trgi` signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on `tim_trgi` signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - `tim_oc1ref` has the same behavior as in PWM mode 1. `tim_oc1refc` is the logical OR between `tim_oc1ref` and `tim_oc2ref`.

1101: Combined PWM mode 2 - `tim_oc1ref` has the same behavior as in PWM mode 2. `tim_oc1refc` is the logical AND between `tim_oc1ref` and `tim_oc2ref`.

1110: Asymmetric PWM mode 1 - `tim_oc1ref` has the same behavior as in PWM mode 1. `tim_oc1refc` outputs `tim_oc1ref` when the counter is counting up, `tim_oc2ref` when it is counting down.

1111: Asymmetric PWM mode 2 - `tim_oc1ref` has the same behavior as in PWM mode 2. `tim_oc1refc` outputs `tim_oc1ref` when the counter is counting up, `tim_oc2ref` when it is counting down.

**Note:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and `CC1S='00'` (the channel is configured in output).

**Note:** In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from "frozen" mode to "PWM" mode and when the output compare mode switches from "force active/inactive" mode to "PWM" mode.

**Note:** On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.
38.6.9 TIMx capture/compare mode register 2 (TIMx_CCMR2) (x = 1, 8)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Input capture mode

**Bit 3 OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=’00’ (the channel is configured in output).*

**Bit 2 OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

**Bits 1:0 CC1S[1:0]**: Capture/compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = ‘0’ in TIMx_CCER).*
Bits 31:16: Reserved, must be kept at reset value.

Bits 15:12: **IC4F[3:0]:** Input capture 4 filter

Bits 11:10: **IC4PSC[1:0]:** Input capture 4 prescaler

Bits 9:8: **CC4S[1:0]:** Capture/compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

- **00:** CC4 channel is configured as output
- **01:** CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
- **10:** CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
- **11:** CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

**Note:** CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bits 7:4: **IC3F[3:0]:** Input capture 3 filter

Bits 3:2: **IC3PSC[1:0]:** Input capture 3 prescaler

Bits 1:0: **CC3S[1:0]:** Capture/compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

- **00:** CC3 channel is configured as output
- **01:** CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
- **10:** CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
- **11:** CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

**Note:** CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

### 38.6.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8)

**Address offset:** 0x01C

**Reset value:** 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

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

**Output compare mode**

Bits 31:25: Reserved, must be kept at reset value.

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

Bit 15: **OC4CE:** Output compare 4 clear enable
Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
   Refer to OC3M[3:0] bit description

   Bit 11 **OC4PE**: Output compare 4 preload enable

   Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection
   This bit-field defines the direction of the channel (input/output) as well as the used input.
   - 00: CC4 channel is configured as output
   - 01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
   - 10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
   - 11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

   *Note:* **CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).**

   Bit 7 **OC3CE**: Output compare 3 clear enable
Bits 16, 6:4 **OC3M[3:0]:** Output compare 3 mode

1540/3119 RM0481 Rev 1

These bits define the behavior of the output reference signal `tim_oc3ref` from which `tim_oc3` and `tim_oc3n` are derived. `tim_oc3ref` is active high whereas `tim_oc3` and `tim_oc3n` active level depends on `CC3P` and `CC3NP` bits.

- **0000:** Frozen - The comparison between the output compare register `TIMx_CCR3` and the counter `TIMx_CNT` has no effect on the outputs. (This mode is used to generate a timing base).
- **0001:** Set channel 3 to active level on match. `tim_oc3ref` signal is forced high when the counter `TIMx_CNT` matches the capture/compare register 3 (`TIMx_CCR3`).
- **0010:** Set channel 3 to inactive level on match. `tim_oc3ref` signal is forced low when the counter `TIMx_CNT` matches the capture/compare register 3 (`TIMx_CCR3`).
- **0011:** Toggle - `tim_oc3ref` toggles when `TIMx_CNT=TIMx_CCR3`.
- **0100:** Force inactive level - `tim_oc3ref` is forced low.
- **0101:** Force active level - `tim_oc3ref` is forced high.
- **0110:** PWM mode 1 - In up-counting, channel 3 is active as long as `TIMx_CNT<TIMx_CCR3` else inactive. In down-counting, channel 3 is inactive as long as `TIMx_CNT>TIMx_CCR3` else active (`tim_oc3ref='1'`).
- **0111:** PWM mode 2 - In up-counting, channel 3 is inactive as long as `TIMx_CNT<TIMx_CCR3` else active. In down-counting, channel 3 is active as long as `TIMx_CNT>TIMx_CCR3` else inactive.
- **1000:** Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on `tim_trgi` signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on `tim_trgi` signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.
- **1001:** Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on `tim_trgi` signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on `tim_trgi` signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.
- **1010:** Pulse on compare: a pulse is generated on `tim_oc3ref` upon `CCR3` match event, as per `PWPRSC[2:0]` and `PW[7:0]` bitfields programming in `TIMxECR`.
- **1011:** Direction output. The `tim_oc3ref` signal is overridden by a copy of the DIR bit.
- **1100:** Combined PWM mode 1 - `tim_oc3ref` has the same behavior as in PWM mode 1. `tim_oc3refc` is the logical OR between `tim_oc3ref` and `tim_oc4ref`.
- **1101:** Combined PWM mode 2 - `tim_oc3ref` has the same behavior as in PWM mode 2. `tim_oc3refc` is the logical AND between `tim_oc3ref` and `tim_oc4ref`.
- **1110:** Asymmetric PWM mode 1 - `tim_oc3ref` has the same behavior as in PWM mode 1. `tim_oc3refc` outputs `tim_oc3ref` when the counter is counting up, `tim_oc4ref` when it is counting down.
- **1111:** Asymmetric PWM mode 2 - `tim_oc3ref` has the same behavior as in PWM mode 2. `tim_oc3refc` outputs `tim_oc3ref` when the counter is counting up, `tim_oc4ref` when it is counting down.

**Note:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and `CC1S='00'` (the channel is configured in output).

**Note:** In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.
38.6.11 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)

Address offset: 0x020
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:22</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 21</td>
<td>CC6P: Capture/compare 6 output polarity</td>
</tr>
<tr>
<td></td>
<td>Refer to CC1P description</td>
</tr>
<tr>
<td>Bit 20</td>
<td>CC6E: Capture/compare 6 output enable</td>
</tr>
<tr>
<td></td>
<td>Refer to CC1E description</td>
</tr>
<tr>
<td>Bits 19:18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 17</td>
<td>CC5P: Capture/compare 5 output polarity</td>
</tr>
<tr>
<td></td>
<td>Refer to CC1P description</td>
</tr>
<tr>
<td>Bit 16</td>
<td>CC5E: Capture/compare 5 output enable</td>
</tr>
<tr>
<td></td>
<td>Refer to CC1E description</td>
</tr>
<tr>
<td>Bit 15</td>
<td>CC4NP: Capture/compare 4 complementary output polarity</td>
</tr>
<tr>
<td></td>
<td>Refer to CC1NP description</td>
</tr>
<tr>
<td>Bit 14</td>
<td>CC4NE: Capture/compare 4 complementary output enable</td>
</tr>
<tr>
<td></td>
<td>Refer to CC1NE description</td>
</tr>
<tr>
<td>Bit 13</td>
<td>CC4P: Capture/compare 4 output polarity</td>
</tr>
<tr>
<td></td>
<td>Refer to CC1P description</td>
</tr>
<tr>
<td>Bit 12</td>
<td>CC4E: Capture/compare 4 output enable</td>
</tr>
<tr>
<td></td>
<td>Refer to CC1E description</td>
</tr>
<tr>
<td>Bit 11</td>
<td>CC3NP: Capture/compare 3 complementary output polarity</td>
</tr>
<tr>
<td></td>
<td>Refer to CC1NP description</td>
</tr>
</tbody>
</table>
Bit 10 **CC3NE**: Capture/compare 3 complementary output enable  
Refer to CC1NE description

Bit 9 **CC3P**: Capture/compare 3 output polarity  
Refer to CC1P description

Bit 8 **CC3E**: Capture/compare 3 output enable  
Refer to CC1E description

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity  
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/compare 2 complementary output enable  
Refer to CC1NE description

Bit 5 **CC2P**: Capture/compare 2 output polarity  
Refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable  
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity  

**CC1 channel configured as output:**  
0: tim_oc1n active high.  
1: tim_oc1n active low.

**CC1 channel configured as input:**  
This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1.  
Refer to CC1P description.

*Note:* This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (channel configured as output).

*Note:* On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable  
0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.  
1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

*Note:* On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.
Bit 1 **CC1P**: Capture/compare 1 output polarity  
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)  
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)  
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.  
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).  
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).  
CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.  
CC1NP=1, CC1P=0: the configuration is reserved, it must not be used.  
**Note:** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).  
**Note:** On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/compare 1 output enable  
0: Capture mode disabled / OC1 is not active (see below)  
1: Capture mode enabled / OC1 signal is output on the corresponding output pin  
When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to Table 395 for details.  
**Note:** On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.
### Table 395. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature

<table>
<thead>
<tr>
<th>Control bits</th>
<th>Output states(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOE bit</td>
<td>OSSI bit</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>Off-State (output enabled with inactive state)</td>
</tr>
<tr>
<td>1</td>
<td>Asynchronously: tim_ocx=CCxP, tim_ocxn=CCxNP (if tim_brk or tim_brk2 is triggered).</td>
</tr>
<tr>
<td>1</td>
<td>Then (this is valid only if tim_brk is triggered), if the clock is present: tim_ocx=OISx and tim_ocxn=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and tim_ocxn both in active state (may cause a short circuit when driving switches in half-bridge configuration). Note: tim_brk2 can only be used if OSSI = OSSR = 1.</td>
</tr>
</tbody>
</table>

---

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

**Note:** The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and the GPIO registers.
38.6.12  TIMx counter (TIMx_CNT)(x = 1, 8)

Address offset: 0x024
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>UIFCPY: UIF copy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.</td>
</tr>
</tbody>
</table>

| Bits 30:16 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>CNT[15:0]: Counter value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Non-dithering mode (DITHEN = 0)</td>
</tr>
<tr>
<td></td>
<td>The register holds the counter value.</td>
</tr>
<tr>
<td></td>
<td>Dithering mode (DITHEN = 1)</td>
</tr>
<tr>
<td></td>
<td>The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.</td>
</tr>
</tbody>
</table>

38.6.13  TIMx prescaler (TIMx_PSC)(x = 1, 8)

Address offset: 0x028
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>PSC[15:0]: Prescaler value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The counter clock frequency (f_{tim_cnt_clk}) is equal to ( \frac{f_{tim_psc_clk}}{PSC[15:0] + 1} ).</td>
</tr>
<tr>
<td></td>
<td>PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).</td>
</tr>
</tbody>
</table>
38.6.14  TIMx auto-reload register (TIMx_ARR)(x = 1, 8)

Address offset: 0x02C
Reset value: 0x0000 FFFF

Bits 31:20  Reserved, must be kept at reset value.
Bits 19:0  **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to the Section 38.3.3: Time-base unit on page 1433 for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
Non-dithering mode (DITHEN = 0)
The register holds the auto-reload value.
Dithering mode (DITHEN = 1)
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

38.6.15  TIMx repetition counter register (TIMx_RCR)(x = 1, 8)

Address offset: 0x030
Reset value: 0x0000

Bits 15:0  **REP[15:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.
When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.
It means in PWM mode (REP+1) corresponds to:
– the number of PWM periods in edge-aligned mode
– the number of half PWM period in center-aligned mode.
38.6.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8)

Address offset: 0x034
Reset value: 0x0000 0000

Bits 31:20 Reserved, must be kept at reset value.
Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0):
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1):
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0):
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1):
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

38.6.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8)

Address offset: 0x038
Reset value: 0x0000 0000

Bits 31:20 Reserved, must be kept at reset value.
Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC1 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0):
The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.
Dithering mode (DITHEN = 1):
The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0):
The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.
Dithering mode (DITHEN = 1):
The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  **CCR2[19:0]: Capture/compare 2 value**

*If channel CC2 is configured as output*: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

**Non-dithering mode (DITHEN = 0)**
The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

**Dithering mode (DITHEN = 1)**
The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

*If channel CC2 is configured as input*: CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

**Non-dithering mode (DITHEN = 0)**
The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

**Dithering mode (DITHEN = 1)**
The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

### 38.6.18  TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8)

**Address offset**: 0x03C

**Reset value**: 0x0000 0000

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CCR3[19:16]</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**CCR3[15:0]:**

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

**Address offset**: 0x03C

**Reset value**: 0x0000 0000

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CCR3[19:16]</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**CCR3[15:0]:**

```
<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  CCR3[19:0]: Capture/compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
   It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
   The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on TIM_OC3 output.
   Non-dithering mode (DITHEN = 0)
   The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.
   Dithering mode (DITHEN = 1)
   The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (TIM_IC3). The TIMx_CCR3 register is read-only and cannot be programmed.
   Non-dithering mode (DITHEN = 0)
   The register holds the capture value in CCR3[15:0]. The CCR3[19:16] bits are reset.
   Dithering mode (DITHEN = 1)
   The register holds the capture in CCR3[19:4]. The CCR3[3:0] bits are reset.

### 38.6.19  TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8)

Address offset: 0x040
Reset value: 0x0000 0000

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

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

CCR4[19:16]:

CCR4[15:0]:

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
38.6.20  TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)

Address offset: 0x044
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Res</td>
<td>BK2BID</td>
<td>BK2BID</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
<td>BK2</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>31      25      19      13      7      1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30      24      18      12      6      0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: As the bits BK2BID/BK2BID/BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSR, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.
Bits 31:30  Reserved, must be kept at reset value.

Bit 29  **BK2BID**: Break2 bidirectional
Refer to BKBID description

Bit 28  **BKBID**: Break bidirectional
0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

*Note:* This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27  **BK2DSRM**: Break2 disarm
Refer to BKDSRM description

Bit 26  **BKDSRM**: Break disarm
0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25  **BK2P**: Break 2 polarity
0: Break input tim_brk2 is active low
1: Break input tim_brk2 is active high

*Note:* This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24  **BK2E**: Break 2 enable
This bit enables the complete break 2 protection (including all sources connected to bk_acth and BKIN sources, as per Figure 384: Break and Break2 circuitry overview).
0: Break2 function disabled
1: Break2 function enabled

*Note:* The BKIN2 must only be used with OSSR = OSSI = 1.

*Note:* This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
Bits 23:20 \textbf{BK2F}[3:0]: Break 2 filter
This bit-field defines the frequency used to sample \textit{tim\_brk2} input and the length of the digital filter applied to \textit{tim\_brk2}. The digital filter is made of an event counter in which \( N \) consecutive events are needed to validate a transition on the output:

- 0000: No filter, \textit{tim\_brk2} acts asynchronously
- 0001: \( f_{\text{SAMPLING}} = f_{\text{tim\_ker\_ck}} \), \( N=2 \)
- 0010: \( f_{\text{SAMPLING}} = f_{\text{tim\_ker\_ck}} \), \( N=4 \)
- 0011: \( f_{\text{SAMPLING}} = f_{\text{tim\_ker\_ck}} \), \( N=8 \)
- 0100: \( f_{\text{SAMPLING}} = f_{\text{DTS}/2} \), \( N=6 \)
- 0101: \( f_{\text{SAMPLING}} = f_{\text{DTS}/2} \), \( N=8 \)
- 0110: \( f_{\text{SAMPLING}} = f_{\text{DTS}/4} \), \( N=6 \)
- 0111: \( f_{\text{SAMPLING}} = f_{\text{DTS}/4} \), \( N=8 \)
- 1000: \( f_{\text{SAMPLING}} = f_{\text{DTS}/8} \), \( N=6 \)
- 1001: \( f_{\text{SAMPLING}} = f_{\text{DTS}/8} \), \( N=8 \)
- 1010: \( f_{\text{SAMPLING}} = f_{\text{DTS}/16} \), \( N=6 \)
- 1011: \( f_{\text{SAMPLING}} = f_{\text{DTS}/16} \), \( N=8 \)
- 1100: \( f_{\text{SAMPLING}} = f_{\text{DTS}/32} \), \( N=6 \)
- 1101: \( f_{\text{SAMPLING}} = f_{\text{DTS}/32} \), \( N=8 \)
- 1110: \( f_{\text{SAMPLING}} = f_{\text{DTS}/32} \), \( N=8 \)

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in \textit{TIMx\_BDTR} register).

Bits 19:16 \textbf{BKF}[3:0]: Break filter
This bit-field defines the frequency used to sample \textit{tim\_brk} input and the length of the digital filter applied to \textit{tim\_brk}. The digital filter is made of an event counter in which \( N \) consecutive events are needed to validate a transition on the output:

- 0000: No filter, \textit{tim\_brk} acts asynchronously
- 0001: \( f_{\text{SAMPLING}} = f_{\text{tim\_ker\_ck}} \), \( N=2 \)
- 0010: \( f_{\text{SAMPLING}} = f_{\text{tim\_ker\_ck}} \), \( N=4 \)
- 0011: \( f_{\text{SAMPLING}} = f_{\text{tim\_ker\_ck}} \), \( N=8 \)
- 0100: \( f_{\text{SAMPLING}} = f_{\text{DTS}/2} \), \( N=6 \)
- 0101: \( f_{\text{SAMPLING}} = f_{\text{DTS}/2} \), \( N=8 \)
- 0110: \( f_{\text{SAMPLING}} = f_{\text{DTS}/4} \), \( N=6 \)
- 0111: \( f_{\text{SAMPLING}} = f_{\text{DTS}/4} \), \( N=8 \)
- 1000: \( f_{\text{SAMPLING}} = f_{\text{DTS}/8} \), \( N=6 \)
- 1001: \( f_{\text{SAMPLING}} = f_{\text{DTS}/8} \), \( N=8 \)
- 1010: \( f_{\text{SAMPLING}} = f_{\text{DTS}/16} \), \( N=6 \)
- 1011: \( f_{\text{SAMPLING}} = f_{\text{DTS}/16} \), \( N=8 \)
- 1100: \( f_{\text{SAMPLING}} = f_{\text{DTS}/32} \), \( N=6 \)
- 1101: \( f_{\text{SAMPLING}} = f_{\text{DTS}/32} \), \( N=8 \)
- 1110: \( f_{\text{SAMPLING}} = f_{\text{DTS}/32} \), \( N=8 \)

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in \textit{TIMx\_BDTR} register).
Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (tim_brk or tim_brk2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled

In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).

See OC/OCN enable description for more details (Section 38.6.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs tim_brk and tim_brk2 is active)

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to bk_acth and BKIN sources, as per Figure 384: Break and Break2 circuitry overview).

0: Break function disabled

1: Break function enabled

*Note:* This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (Section 38.6.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

*Note:* This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 10 **OSSI**: Off-state selection for idle mode
This bit is used when MOE=0 due to a break event or by a software write, on channels configured as outputs.

See OC/OCN enable description for more details (Section 38.6.11: TIMx capture/compare enable register (TIMx_CCRER)(x = 1, 8)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).
1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

**Note**: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration
These bits offer a write protection against software errors.
00: LOCK OFF - No bit is write protected.
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBED/BK2BED/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCRER register, as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

**Note**: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup
This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x t\_d\_tg with t\_d\_tg=\_D\_T\_S.
DTG[7:5]=10x => DT=(64+DTG[5:0])xT\_d\_tg with T\_d\_tg=2xT\_D\_T\_S.
DTG[7:5]=110 => DT=(32+DTG[4:0])xT\_d\_tg with T\_d\_tg=8xT\_D\_T\_S.
DTG[7:5]=111 => DT=(32+DTG[4:0])xT\_d\_tg with T\_d\_tg=16xT\_D\_T\_S.

Example if T\_D\_T\_S=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 us to 31750 ns by 250 ns steps,
32 us to 63 us by 1 us steps,
64 us to 126 us by 2 us steps

**Note**: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

### 38.6.21 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8)

Address offset: 0x048

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>CCR5[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

1554/3119
38.6.22 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8)

Address offset: 0x04C
Reset value: 0x0000 0000
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  **CCR6[19:0]: Capture/compare 6 value**

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register
(bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when
an update event occurs.
The active capture/compare register contains the value to be compared to the counter
TIMx_CNT and signaled on tim_oc6 output.

**Non-dithering mode (DITHEN = 0)**
The register holds the compare value in CCR6[15:0]. The CCR6[19:16] bits are reset.

**Dithering mode (DITHEN = 1)**
The register holds the integer part in CCR6[19:4]. The CCR6[3:0] bitfield contains the
dithered part.

### 38.6.23  TIMx capture/compare mode register 3 (TIMx_CCMR3)  
(x = 1, 8)

Address offset: 0x050
Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

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

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

Bits 31:25  Reserved, must be kept at reset value.

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

Bit 15  **OC6CE: Output compare 6 clear enable**

Bits 24, 14:12  **OC6M[3:0]: Output compare 6 mode**

Bit 11  **OC6PE: Output compare 6 preload enable**

Bit 10  **OC6FE: Output compare 6 fast enable**

Bits 9:8  Reserved, must be kept at reset value.

Bit 7  **OC5CE: Output compare 5 clear enable**

Bits 16, 6:4  **OC5M[3:0]: Output compare 5 mode**

Bit 3  **OC5PE: Output compare 5 preload enable**

Bit 2  **OC5FE: Output compare 5 fast enable**

Bits 1:0  Reserved, must be kept at reset value.
38.6.24  TIMx timer deadtime register 2 (TIMx_DTR2)(x = 1, 8)

Address offset: 0x054
Reset value: 0x0000 0000

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

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

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

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

Bit 17  **DTPE**: Deadtime preload enable
0: Deadtime value is not preloaded
1: Deadtime value preload is enabled

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 16  **DTAE**: Deadtime asymmetric enable
0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register
1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*

Bits 15:8  Reserved, must be kept at reset value.

Bits 7:0  **DTGF[7:0]**: Dead-time falling edge generator setup
This bit-field defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5]=0xx => DTF=DTGF[7:0]x T_{dtg} with T_{dtg}=T_{DTS}.
DTGF[7:5]=10x => DTF=(64+DTGF[5:0])x T_{dtg} with T_{dtg}=2x T_{DTS}.
DTGF[7:5]=110 => DTF=(32+DTGF[4:0])x T_{dtg} with T_{dtg}=8x T_{DTS}.
DTGF[7:5]=111 => DTF=(32+DTGF[4:0])x T_{dtg} with T_{dtg}=16x T_{DTS}.

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 us to 31750 ns by 250 ns steps,
32 us to 63us by 1 us steps,
64 us to 126 us by 2 us steps

*Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*
38.6.25 TIMx timer encoder control register (TIMx_ECR)(x = 1, 8)

Address offset: 0x058
Reset value: 0x0000 0000

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

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

\[ t_{PWG} = (2^{PWPRSC[2:0]}) \times t_{tim.ker.ck} \]

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

\[ t_{PW} = PW[7:0] \times t_{PWG} \]

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

*Note: IPOS[1] bit is not significant*

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter
Bits 4:3 **IBLK[1:0]**: Index blanking
This bit indicates if the Index event is conditioned by the `tim_ti3` or `tim_ti4` input
00: Index always active
01: Index disabled when `tim_ti3` input is active, as per CC3P bitfield
10: Index disabled when `tim_ti4` input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction
This bit indicates in which direction the Index event resets the counter.
00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable
This bit indicates if the Index event resets the counter.
0: Index disabled
1: Index enabled

### 38.6.26 TIMx timer input selection register (TIMx_TISEL)(x = 1, 8)

Address offset: 0x05C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI4SEL[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI3SEL[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI2SEL[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI1SEL[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects `tim_ti4[15:0]` input
0000: `tim_ti4_in0`: TIMx_CH4
0001: `tim_ti4_in1`
...
1111: `tim_ti4_in15`
Refer to Section 38.3.2: TIM1/TIM8 pins and internal signals for interconnects list.

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

Bits 19:16 **TI3SEL[3:0]**: Selects `tim_ti3[15:0]` input
0000: `tim_ti3_in0`: TIMx_CH2
0001: `tim_ti3_in1`
...
1111: `tim_ti3_in15`
Refer to Section 38.3.2: TIM1/TIM8 pins and internal signals for interconnects list.

Bits 15:12 Reserved, must be kept at reset value.
38.6.27 TIMx alternate function option register 1 (TIMx_AF1)(x = 1, 8)

Address offset: 0x060
Reset value: 0x0000 0001

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

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

Bits 17:14 ETRSEL[3:0]: etr_in source selection
These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to Section 38.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 BKCMP4P: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: tim_brk_cmp4 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

- This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.
  - 0: tim_brk_cmp3 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
  - 1: tim_brk_cmp3 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

- This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.
  - 0: tim_brk_cmp2 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
  - 1: tim_brk_cmp2 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

- This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.
  - 0: tim_brk_cmp1 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
  - 1: tim_brk_cmp1 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 9 **BKINP**: TIMx_BKIN input polarity

- This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
  - 0: TIMx_BKIN input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
  - 1: TIMx_BKIN input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

- This bit enables the tim_brk_cmp8 for the timer’s tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.
  - 0: tim_brk_cmp8 input disabled
  - 1: tim_brk_cmp8 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

- This bit enables the tim_brk_cmp7 for the timer’s tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.
  - 0: tim_brk_cmp7 input disabled
  - 1: tim_brk_cmp7 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

- This bit enables the tim_brk_cmp6 for the timer’s tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.
  - 0: tim_brk_cmp6 input disabled
  - 1: tim_brk_cmp6 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*
Bit 5 **BKCMP5E**: tim_brk_cmp5 enable
- This bit enables the tim_brk_cmp5 for the timer’s tim_brk input. tim_brk_cmp5 output is ‘ORed’ with the other tim_brk sources.
  - 0: tim_brk_cmp5 input disabled
  - 1: tim_brk_cmp5 input enabled

  **Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable
- This bit enables the tim_brk_cmp4 for the timer’s tim_brk input. tim_brk_cmp4 output is ‘ORed’ with the other tim_brk sources.
  - 0: tim_brk_cmp4 input disabled
  - 1: tim_brk_cmp4 input enabled

  **Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable
- This bit enables the tim_brk_cmp3 for the timer’s tim_brk input. tim_brk_cmp3 output is ‘ORed’ with the other tim_brk sources.
  - 0: tim_brk_cmp3 input disabled
  - 1: tim_brk_cmp3 input enabled

  **Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable
- This bit enables the tim_brk_cmp2 for the timer’s tim_brk input. tim_brk_cmp2 output is ‘ORed’ with the other tim_brk sources.
  - 0: tim_brk_cmp2 input disabled
  - 1: tim_brk_cmp2 input enabled

  **Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable
- This bit enables the tim_brk_cmp1 for the timer’s tim_brk input. tim_brk_cmp1 output is ‘ORed’ with the other tim_brk sources.
  - 0: tim_brk_cmp1 input disabled
  - 1: tim_brk_cmp1 input enabled

  **Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable
- This bit enables the TIMx_BKIN alternate function input for the timer’s tim_brk input. TIMx_BKIN input is ‘ORed’ with the other tim_brk sources.
  - 0: TIMx_BKIN input disabled
  - 1: TIMx_BKIN input enabled

  **Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

**Note**: Refer to Section 38.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.
38.6.28 TIMx alternate function register 2 (TIMx_AF2)(x = 1, 8)

Address offset: 0x064
Reset value: 0x0000 0001

| Bit 31:19 | Reserved, must be kept at reset value. |
| Bit 18:16 | OCRSEL[2:0]: ocref_clr source selection |
|           | These bits select the ocref_clr input source. |
|           | 000: tim_ocref_clr0 |
|           | 001: tim_ocref_clr1 |
|           | ... |
|           | 111: tim_ocref_clr7 |
|           | Refer to Section 38.3.2: TIM1/TIM8 pins and internal signals for product specific information. |
| Note:     | These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register). |

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

Bit 13 BK2CMP4P: tim_brk2_cmp4 input polarity
This bit selects the tim_brk2_cmp4 input sensitivity. It must be programmed together with the BK2P polarity bit.
0: tim_brk2_cmp4 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)
1: tim_brk2_cmp4 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 BK2CMP3P: tim_brk2_cmp3 input polarity
This bit selects the tim_brk2_cmp3 input sensitivity. It must be programmed together with the BK2P polarity bit.
0: tim_brk2_cmp3 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)
1: tim_brk2_cmp3 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 BK2CMP2P: tim_brk2_cmp2 input polarity
This bit selects the tim_brk2_cmp2 input sensitivity. It must be programmed together with the BK2P polarity bit.
0: tim_brk2_cmp2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)
1: tim_brk2_cmp2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 10 **BK2CMP1**: tim_brk2_cmp1 input polarity

This bit selects the tim_brk2_cmp1 input sensitivity. It must be programmed together with the BK2P polarity bit.

- 0: tim_brk2_cmp1 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)
- 1: tim_brk2_cmp1 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

**Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BK2INP**: TIMx_BKIN2 input polarity

This bit selects the TIMx_BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

- 0: TIMx_BKIN2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)
- 1: TIMx_BKIN2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

**Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2CMP8**: tim_brk2_cmp8 enable

This bit enables the tim_brk2_cmp8 for the timer’s tim_brk2 input. tim_brk2_cmp8 output is 'ORed' with the other tim_brk2 sources.

- 0: tim_brk2_cmp8 input disabled
- 1: tim_brk2_cmp8 input enabled

**Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BK2CMP7**: tim_brk2_cmp7 enable

This bit enables the tim_brk2_cmp7 for the timer’s tim_brk2 input. tim_brk2_cmp7 output is 'ORed' with the other tim_brk2 sources.

- 0: tim_brk2_cmp7 input disabled
- 1: tim_brk2_cmp7 input enabled

**Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BK2CMP6**: tim_brk2_cmp6 enable

This bit enables the tim_brk2_cmp6 for the timer’s tim_brk2 input. tim_brk2_cmp6 output is 'ORed' with the other tim_brk2 sources.

- 0: tim_brk2_cmp6 input disabled
- 1: tim_brk2_cmp6 input enabled

**Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BK2CMP5**: tim_brk2_cmp5 enable

This bit enables the tim_brk2_cmp5 for the timer’s tim_brk2 input. tim_brk2_cmp5 output is 'ORed' with the other tim_brk2 sources.

- 0: tim_brk2_cmp5 input disabled
- 1: tim_brk2_cmp5 input enabled

**Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BK2CMP4**: tim_brk2_cmp4 enable

This bit enables the tim_brk2_cmp4 for the timer’s tim_brk2 input. tim_brk2_cmp4 output is 'ORed' with the other tim_brk2 sources.

- 0: tim_brk2_cmp4 input disabled
- 1: tim_brk2_cmp4 input enabled

**Note**: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 3 **BK2CMP3**: tim_brk2_cmp3 enable  
This bit enables the tim_brk2_cmp3 for the timer’s tim_brk2 input. tim_brk2_cmp3 output is 'ORed' with the other tim_brk2 sources.  
0: tim_brk2_cmp3 input disabled  
1: tim_brk2_cmp3 input enabled  
*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 2 **BK2CMP2**: tim_brk2_cmp2 enable  
This bit enables the tim_brk2_cmp2 for the timer’s tim_brk2 input. tim_brk2_cmp2 output is 'ORed' with the other tim_brk2 sources.  
0: tim_brk2_cmp2 input disabled  
1: tim_brk2_cmp2 input enabled  
*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 1 **BK2CMP1**: tim_brk2_cmp1 enable  
This bit enables the tim_brk2_cmp1 for the timer’s tim_brk2 input. tim_brk2_cmp1 output is 'ORed' with the other tim_brk2 sources.  
0: tim_brk2_cmp1 input disabled  
1: tim_brk2_cmp1 input enabled  
*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 0 **BK2INE**: TIMx_BKIN2 input enable  
This bit enables the TIMx_BKIN2 alternate function input for the timer’s tim_brk2 input. TIMx_BKIN2 input is 'ORed' with the other tim_brk2 sources.  
0: TIMx_BKIN2 input disabled  
1: TIMx_BKIN2 input enabled  
*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

*Note: Refer to Section 38.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.*

### 38.6.29 TIMx DMA control register (TIMx_DCR)(x = 1, 8)

Address offset: 0x3DC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:16  **DBSS[3:0]: DMA burst source selection**
This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

- 0000: Reserved
- 0001: Update
- 0010: CC1
- 0011: CC2
- 0100: CC3
- 0101: CC4
- 0110: COM
- 0111: Trigger
- Others: reserved

Bits 15:13  Reserved, must be kept at reset value.

Bits 12:8  **DBL[4:0]: DMA burst length**
This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

- 00000: 1 transfer
- 00001: 2 transfers
- 00010: 3 transfers
- ...
- 11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

- If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:
  \[(\text{TIMx\_CR1 address}) + \text{DBA} + (\text{DMA index})\]
  where DMA index = DBL

In this example, 7 bytes are added to (TIMx\_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx\_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

- If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.
- If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on.

So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5  Reserved, must be kept at reset value.

Bits 4:0  **DBA[4:0]: DMA base address**
This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register.

Example:
- 00000: TIMx\_CR1
- 00001: TIMx\_CR2
- 00010: TIMx\_SMCR
- ...

38.6.30  **TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8)**

Address offset: 0x3E0  
Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]:** DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4 where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

38.6.31  **TIMx register map**

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>TIMx_CR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>TIMx_CR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>TIMx_SMCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C</td>
<td>TIMx_DIER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>TIMx_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>TIMx_EGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 396. TIMx register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>TIMx_CR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>TIMx_CR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>TIMx_SMCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C</td>
<td>TIMx_DIER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>TIMx_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>TIMx_EGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

STI
### Table 396. TIMx register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset 16 (Little Endian)</th>
<th>Offset 16 (Big Endian)</th>
<th>Offset 8 (Little Endian)</th>
<th>Offset 8 (Big Endian)</th>
<th>Offset 4 (Little Endian)</th>
<th>Offset 4 (Big Endian)</th>
<th>Offset 2 (Little Endian)</th>
<th>Offset 2 (Big Endian)</th>
<th>Offset 1 (Little Endian)</th>
<th>Offset 1 (Big Endian)</th>
<th>Offset 0 (Little Endian)</th>
<th>Offset 0 (Big Endian)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x018</td>
<td>TIMx_CCMR1</td>
<td>TIM0_CCMR0</td>
<td>TIM1_CCMR1</td>
<td>TIM2_CCMR0</td>
<td>TIM3_CCMR1</td>
<td>TIM0_CCMR1</td>
<td>TIM1_CCMR1</td>
<td>TIM2_CCMR0</td>
<td>TIM3_CCMR1</td>
<td>TIM0_CCMR1</td>
<td>TIM1_CCMR1</td>
<td>TIM2_CCMR0</td>
<td>TIM3_CCMR1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x01C</td>
<td>TIMx_CCMR1</td>
<td>TIM0_CCMR1</td>
<td>TIM1_CCMR1</td>
<td>TIM2_CCMR0</td>
<td>TIM3_CCMR1</td>
<td>TIM0_CCMR1</td>
<td>TIM1_CCMR1</td>
<td>TIM2_CCMR0</td>
<td>TIM3_CCMR1</td>
<td>TIM0_CCMR1</td>
<td>TIM1_CCMR1</td>
<td>TIM2_CCMR0</td>
<td>TIM3_CCMR1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x020</td>
<td>TIMx_CCMR2</td>
<td>TIM0_CCMR2</td>
<td>TIM1_CCMR2</td>
<td>TIM2_CCMR2</td>
<td>TIM3_CCMR2</td>
<td>TIM0_CCMR2</td>
<td>TIM1_CCMR2</td>
<td>TIM2_CCMR2</td>
<td>TIM3_CCMR2</td>
<td>TIM0_CCMR2</td>
<td>TIM1_CCMR2</td>
<td>TIM2_CCMR2</td>
<td>TIM3_CCMR2</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x024</td>
<td>TIMx_CNT</td>
<td>TIM0_CNT</td>
<td>TIM1_CNT</td>
<td>TIM2_CNT</td>
<td>TIM3_CNT</td>
<td>TIM0_CNT</td>
<td>TIM1_CNT</td>
<td>TIM2_CNT</td>
<td>TIM3_CNT</td>
<td>TIM0_CNT</td>
<td>TIM1_CNT</td>
<td>TIM2_CNT</td>
<td>TIM3_CNT</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x028</td>
<td>TIMx_PSC</td>
<td>TIM0_PSC</td>
<td>TIM1_PSC</td>
<td>TIM2_PSC</td>
<td>TIM3_PSC</td>
<td>TIM0_PSC</td>
<td>TIM1_PSC</td>
<td>TIM2_PSC</td>
<td>TIM3_PSC</td>
<td>TIM0_PSC</td>
<td>TIM1_PSC</td>
<td>TIM2_PSC</td>
<td>TIM3_PSC</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x02C</td>
<td>TIMx_ARR</td>
<td>TIM0_ARR</td>
<td>TIM1_ARR</td>
<td>TIM2_ARR</td>
<td>TIM3_ARR</td>
<td>TIM0_ARR</td>
<td>TIM1_ARR</td>
<td>TIM2_ARR</td>
<td>TIM3_ARR</td>
<td>TIM0_ARR</td>
<td>TIM1_ARR</td>
<td>TIM2_ARR</td>
<td>TIM3_ARR</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x030</td>
<td>TIMx_RCR</td>
<td>TIM0_RCR</td>
<td>TIM1_RCR</td>
<td>TIM2_RCR</td>
<td>TIM3_RCR</td>
<td>TIM0_RCR</td>
<td>TIM1_RCR</td>
<td>TIM2_RCR</td>
<td>TIM3_RCR</td>
<td>TIM0_RCR</td>
<td>TIM1_RCR</td>
<td>TIM2_RCR</td>
<td>TIM3_RCR</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x034</td>
<td>TIMx_CCR1</td>
<td>TIM0_CCR1</td>
<td>TIM1_CCR1</td>
<td>TIM2_CCR1</td>
<td>TIM3_CCR1</td>
<td>TIM0_CCR1</td>
<td>TIM1_CCR1</td>
<td>TIM2_CCR1</td>
<td>TIM3_CCR1</td>
<td>TIM0_CCR1</td>
<td>TIM1_CCR1</td>
<td>TIM2_CCR1</td>
<td>TIM3_CCR1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x038</td>
<td>TIMx_CCR2</td>
<td>TIM0_CCR2</td>
<td>TIM1_CCR2</td>
<td>TIM2_CCR2</td>
<td>TIM3_CCR2</td>
<td>TIM0_CCR2</td>
<td>TIM1_CCR2</td>
<td>TIM2_CCR2</td>
<td>TIM3_CCR2</td>
<td>TIM0_CCR2</td>
<td>TIM1_CCR2</td>
<td>TIM2_CCR2</td>
<td>TIM3_CCR2</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x03C</td>
<td>TIMx_CCR3</td>
<td>TIM0_CCR3</td>
<td>TIM1_CCR3</td>
<td>TIM2_CCR3</td>
<td>TIM3_CCR3</td>
<td>TIM0_CCR3</td>
<td>TIM1_CCR3</td>
<td>TIM2_CCR3</td>
<td>TIM3_CCR3</td>
<td>TIM0_CCR3</td>
<td>TIM1_CCR3</td>
<td>TIM2_CCR3</td>
<td>TIM3_CCR3</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x040</td>
<td>TIMx_CCR4</td>
<td>TIM0_CCR4</td>
<td>TIM1_CCR4</td>
<td>TIM2_CCR4</td>
<td>TIM3_CCR4</td>
<td>TIM0_CCR4</td>
<td>TIM1_CCR4</td>
<td>TIM2_CCR4</td>
<td>TIM3_CCR4</td>
<td>TIM0_CCR4</td>
<td>TIM1_CCR4</td>
<td>TIM2_CCR4</td>
<td>TIM3_CCR4</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x044</td>
<td>TIMx_BDTR</td>
<td>TIM0_BDTR</td>
<td>TIM1_BDTR</td>
<td>TIM2_BDTR</td>
<td>TIM3_BDTR</td>
<td>TIM0_BDTR</td>
<td>TIM1_BDTR</td>
<td>TIM2_BDTR</td>
<td>TIM3_BDTR</td>
<td>TIM0_BDTR</td>
<td>TIM1_BDTR</td>
<td>TIM2_BDTR</td>
<td>TIM3_BDTR</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Refer to Section 2.3 on page 111 for the register boundary addresses.
39 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

39.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in Section 39.4.23: Timer synchronization.

39.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit or 32-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management
### 39.3 TIM2/TIM3/TIM4/TIM5 implementation

Table 397. STM32H563/H573 and STM32H562 general purpose timers

<table>
<thead>
<tr>
<th>Timer instance</th>
<th>TIM2</th>
<th>TIM3</th>
<th>TIM4</th>
<th>TIM5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>32-bit</td>
<td>16-bit</td>
<td>16-bit</td>
<td>32-bit</td>
</tr>
<tr>
<td>OCREF clear selection Sources</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sources</td>
<td>tim_etrf</td>
<td>tim_etrf</td>
<td>tim_etrf</td>
<td>tim_etrf</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
39.4 TIM2/TIM3/TIM4/TIM5 functional description

39.4.1 Block diagram

Figure 427. General-purpose timer block diagram

1. This feature is not available on all timers, refer to the Section 39.3: TIM2/TIM3/TIM4/TIM5 implementation.
39.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals

Table 398 and Table 399 in this section summarize the TIM inputs and outputs.

### Table 398. TIM input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_CH1</td>
<td>Input/Output</td>
<td>Timer multi-purpose channels. Each channel can be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.</td>
</tr>
<tr>
<td>TIM_CH2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_ETR</td>
<td>Input</td>
<td>External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etru_in prescaler is used.</td>
</tr>
</tbody>
</table>

### Table 399. TIM internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti1_in[15:0]</td>
<td>Input</td>
<td>Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.</td>
</tr>
<tr>
<td>tim_ti2_in[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_ti3_in[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_ti4_in[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_etru[15:0]</td>
<td>Input</td>
<td>External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etru_in prescaler is used.</td>
</tr>
<tr>
<td>tim_itr[15:0]</td>
<td>Input</td>
<td>Internal trigger input bus. These inputs can be used for the slave mode controller or as an input clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_trgo</td>
<td>Output</td>
<td>Internal trigger output. This trigger can trigger other on-chip peripherals.</td>
</tr>
<tr>
<td>tim_ocref clr[7:0]</td>
<td>Input</td>
<td>Timer tim_ocref clr input bus. These inputs can be used to clear the tim_ocref signals, typically for hardware cycle-by-cycle pulse width control.</td>
</tr>
<tr>
<td>tim_pclk</td>
<td>Input</td>
<td>Timer APB clock.</td>
</tr>
<tr>
<td>tim_ker_ck</td>
<td>Input</td>
<td>Timer kernel clock</td>
</tr>
</tbody>
</table>
Table 399. TIM internal input/output signals (continued)

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_it</td>
<td>Output</td>
<td>Global Timer interrupt, gathering capture/compare, update and break trigger requests.</td>
</tr>
<tr>
<td>tim_cc1_dma</td>
<td>Output</td>
<td>Timer capture / compare 1..4 dma requests.</td>
</tr>
<tr>
<td>tim_cc2_dma</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_cc3_dma</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_cc4_dma</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim upd_dma</td>
<td>Output</td>
<td>Timer update dma request.</td>
</tr>
<tr>
<td>tim_trg_dma</td>
<td>Output</td>
<td>Timer trigger dma request.</td>
</tr>
</tbody>
</table>

Table 400, Table 401, Table 402 and Table 403 are listing the sources connected to the tim_ti[4:1] input multiplexers.

Table 400. Interconnect to the tim_ti1 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti1 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM2</td>
</tr>
<tr>
<td>tim_ti1_in0</td>
<td>TIM2_CH1</td>
</tr>
<tr>
<td>tim_ti1_in1</td>
<td>eth_ptp_pps_o</td>
</tr>
<tr>
<td>tim_ti1_in[15:2]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 401. Interconnect to the tim_ti2 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti2 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM2</td>
</tr>
<tr>
<td>tim_ti2_in0</td>
<td>TIM2_CH2</td>
</tr>
<tr>
<td>tim_ti2_in[15:1]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 402. Interconnect to the tim_ti3 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti3 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM2</td>
</tr>
<tr>
<td>tim_ti3_in0</td>
<td>TIM2_CH3</td>
</tr>
<tr>
<td>tim_ti3_in[15:1]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
### Table 403. Interconnect to the tim_ti4 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti4 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM2</td>
</tr>
<tr>
<td>tim_ti4_in0</td>
<td>TIM2_CH4</td>
</tr>
<tr>
<td>tim_ti4_in[15:1]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

*Table 403 lists the internal sources connected to the tim_ti4 input multiplexer.*

### Table 404. TIMx internal trigger connection

<table>
<thead>
<tr>
<th>TIMx</th>
<th>TIM2</th>
<th>TIM3</th>
<th>TIM4</th>
<th>TIM5</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_itr0</td>
<td>tim1_trgo</td>
<td>tim1_trgo</td>
<td>tim1_trgo</td>
<td>tim1_trgo</td>
</tr>
<tr>
<td>tim_itr1</td>
<td>Reserved</td>
<td>tim2_trgo</td>
<td>tim2_trgo</td>
<td>tim2_trgo</td>
</tr>
<tr>
<td>tim_itr2</td>
<td>tim3_trgo</td>
<td>Reserved</td>
<td>tim3_trgo</td>
<td>tim3_trgo</td>
</tr>
<tr>
<td>tim_itr3</td>
<td>tim4_trgo</td>
<td>tim4_trgo</td>
<td>Reserved</td>
<td>tim4_trgo</td>
</tr>
<tr>
<td>tim_itr4</td>
<td>tim5_trgo</td>
<td>tim5_trgo</td>
<td>tim5_trgo</td>
<td>Reserved</td>
</tr>
<tr>
<td>tim_itr5</td>
<td>tim8_trgo</td>
<td>tim8_trgo</td>
<td>tim8_trgo</td>
<td>tim8_trgo</td>
</tr>
<tr>
<td>tim_itr6</td>
<td>tim12_trgo</td>
<td>tim12_trgo</td>
<td>tim12_trgo</td>
<td>tim12_trgo</td>
</tr>
<tr>
<td>tim_itr7</td>
<td>tim13_oc1</td>
<td>tim13_oc1</td>
<td>tim13_oc1</td>
<td>tim13_oc1</td>
</tr>
<tr>
<td>tim_itr8</td>
<td>tim14_oc1</td>
<td>tim14_oc1</td>
<td>tim14_oc1</td>
<td>tim14_oc1</td>
</tr>
<tr>
<td>tim_itr9</td>
<td>tim15_trgo</td>
<td>tim15_trgo</td>
<td>tim15_trgo</td>
<td>tim15_trgo</td>
</tr>
<tr>
<td>tim_itr10</td>
<td>tim16_oc1</td>
<td>tim16_oc1</td>
<td>tim16_oc1</td>
<td>tim16_oc1</td>
</tr>
<tr>
<td>tim_itr11</td>
<td>tim17_oc1</td>
<td>tim17_oc1</td>
<td>tim17_oc1</td>
<td>tim17_oc1</td>
</tr>
<tr>
<td>tim_itr12</td>
<td>USBSOF</td>
<td>Reserved</td>
<td>Reserved</td>
<td>USBSOF</td>
</tr>
<tr>
<td>tim_itr[15:13]</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 404 lists the internal sources connected to the tim_itr input multiplexer.*

### Table 405. Interconnect to the tim_etr input multiplexer

<table>
<thead>
<tr>
<th>Timer external trigger input signal</th>
<th>Timer external trigger signals assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM2</td>
</tr>
<tr>
<td>tim_etr0</td>
<td>TIM2_ETR</td>
</tr>
<tr>
<td>tim_etr1</td>
<td>Reserved</td>
</tr>
<tr>
<td>tim_etr2</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
### 39.4.3 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:
- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.
**Prescaler description**

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 428* and *Figure 429* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

**Figure 428. Counter timing diagram with prescaler division change from 1 to 2**

![Counter timing diagram with prescaler division change from 1 to 2](image-url)
### 39.4.4 Counter modes

#### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.
Figure 430. Counter timing diagram, internal clock divided by 1

Figure 431. Counter timing diagram, internal clock divided by 2
Figure 432. Counter timing diagram, internal clock divided by 4

Figure 433. Counter timing diagram, internal clock divided by N
Figure 434. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

- **tim_psc_ck**
- **CEN**
- **tim_cnt_ck**
- **Counter register**: 31 32 33 34 35 36 00 01 02 03 04 05 06 07
- **Counter overflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**
- **Auto-reload preload register**: FF 36

Write a new value in TIMx_ARR
**Downcounting mode**

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

An Update event can be generated at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0.

However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn’t change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.
When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

**Figure 436. Counter timing diagram, internal clock divided by 1**
**Figure 437. Counter timing diagram, internal clock divided by 2**

- `tim_psc_ck`
- `CEN`
- `tim_cnt_ck`
- Counter register: 0002 0001 0000 0036 0035 0034 0033
- Counter underflow
- Update event (UEV)
- Update interrupt flag (UIF)

**Figure 438. Counter timing diagram, internal clock divided by 4**

- `tim_psc_ck`
- `CEN`
- `tim_cnt_ck`
- Counter register: 0001 0000 0000 0001
- Counter underflow
- Update event (UEV)
- Update interrupt flag (UIF)
Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-
reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.
Figure 441. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6

1. Here, center-aligned mode 1 is used (for more details refer to Section 39.5.1: TIMx control register 1 (TIMx_CR1)(x = 2 to 5) on page 1648).

Figure 442. Counter timing diagram, internal clock divided by 2
Figure 443. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 444. Counter timing diagram, internal clock divided by N
Figure 445. Counter timing diagram, Update event with ARPE=1 (counter underflow)
39.4.5 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, Timer 1 can be configured to act as a prescaler for Timer 2. Refer to "Using one timer as prescaler for another timer on page 1640" for more details.

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS=000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 447 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.
External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 448. tim_ti2 external clock connection example

1. Codes ranging from 01000 to 11111: tim_itr[15:0].

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:
1. Select the proper `ti2_in[15:0]` source (internal or external) with the `TI2SEL[3:0]` bits in the `TIMx_TISEL` register.
2. Configure channel 2 to detect rising edges on the `ti2_in` input by writing `CC2S= '01` in the `TIMx_CCMR1` register.
3. Configure the input filter duration by writing the `IC2F[3:0]` bits in the `TIMx_CCMR1` register (if no filter is needed, keep `IC2F=0000`).

**Note:** The capture prescaler is not used for triggering, so it does not need to be configured.
4. Select rising edge polarity by writing `CC2P=0` and `CC2NP=0` and `CC2NP=0` in the `TIMx_CCER` register.
5. Configure the timer in external clock mode 1 by writing `SMS=111` in the `TIMx_SMCR` register.
6. Select `ti2` as the input source by writing `TS=00110` in the `TIMx_SMCR` register.
7. Enable the counter by writing `CEN=1` in the `TIMx_CR1` register.

When a rising edge occurs on `ti2`, the counter counts once and the `TIF` flag is set.

The delay between the rising edge on `ti2` and the actual clock of the counter is due to the resynchronization circuit on `ti2` input.

**Figure 449. Control circuit in external clock mode 1**

External clock source mode 2

This mode is selected by writing `ECE=1` in the `TIMx_SMCR` register.

The counter can count at each rising or falling edge on the external trigger input `etr_in`.

**Figure 450** gives an overview of the external trigger input block.
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

1. Select the proper tim_etr_in source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register.
2. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register.
4. Select rising edge detection on the tim_etr_in by writing ETP=0 in the TIMx_SMCR register.
5. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrf signal. As a consequence, the maximum frequency that can be correctly captured by the counter is at most ¼ of TIMxCLK frequency.

When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.
39.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).
The output stage generates an intermediate waveform which is then used for reference: `tim_ocxref` (active high). The polarity acts at the end of the chain.

**Figure 453. Capture/compare channel 1 main circuit**

**Figure 454. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)**

1. Available on some instances only. If not available, `tim_etrf` is directly connected to `tim_ocref_clr_int`.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.
In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 39.4.7 Input capture mode

In input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_t1 input rises. To do this, use the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_t1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the needed input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let’s imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on tim_t1 when 8 consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_t1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCR register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:
- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

**Note:** IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.
39.4.8 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges)

This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tix input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two T1xFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulse width of a PWM signal applied on tim_t1 can be measured using the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TIMx_TISEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_t1 selected).
3. Select the active polarity for tim_t1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to ‘0’ and the CC1NP bit to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_t1 selected).
5. Select the active polarity for tim_t1fp2 (used for capture in TIMx_CCR2): write the CC2P bit to ‘1’ and the CC2NP bit to ‘0’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_t1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.
39.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP=0 (tim_ocx active high) => tim_ocx is forced to high level.

tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

39.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:
- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP)
bit in the TIMx_CCER register). The output pin can keep its level (OCxM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example:
   a) Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   b) Write OCxPE = 0 to disable preload register
   c) Write CCxP = 0 to select active high polarity
   d) Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 456.
39.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per TIMx output) by writing 110 (PWM mode 1) or ‘111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). The tim_ocref_clr can be cleared by an external event through the tim_etr_in or the tim_ocref_clr signals. In this case the tim_ocref_clr signal is asserted only:

- After a compare match event
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM=’000) to one of the PWM modes (OCxM=’110 or ‘111). This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.
**PWM edge-aligned mode**

**Upcounting configuration**

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode on page 1578*.

In the following example, we consider PWM mode 1. The reference PWM signal `tim_ocxref` is high as long as `TIMx_CNT < TIMx_CCRx` else it becomes low. If the compare value in `TIMx_CCRx` is greater than the auto-reload value (in `TIMx_ARR`) then `tim_ocxref` is held at '1'. If the compare value is 0 then `tim_ocxref` is held at '0'. *Figure 457* shows some edge-aligned PWM waveforms in an example where `TIMx_ARR`=8.

*Figure 457. Edge-aligned PWM waveforms (ARR=8)*

![Counter register diagram](diagram)

**Downcounting configuration**

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to *Downcounting mode on page 1582*.

In PWM mode 1, the reference signal `tim_ocxref` is low as long as `TIMx_CNT > TIMx_CCRx` else it becomes high. If the compare value in `TIMx_CCRx` is greater than the auto-reload value in `TIMx_ARR`, then `tim_ocxref` is held at 100%. PWM is not possible in this mode.

**PWM center-aligned mode**

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00 (all the remaining configurations having the same effect on the `tim_ocxref`/`tim_ocx` signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit
(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to Center-aligned mode (up/down counting) on page 1585.

Figure 458 shows some center-aligned PWM waveforms in an example where:
- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Figure 458. Center-aligned PWM waveforms (ARR=8)

Hints on using center-aligned mode:
- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit
in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.

- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The Figure 459 below presents the dithering principle applied to 4 consecutive PWM cycles.

![Figure 459. Dithering principle](MS45752V1)

When the dithering mode is enabled, the register coding is changed as following (see Figure 460 for example):

- The 4 LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted by 4 places and are coding for the base value. In 16-bit mode, the 16-bit format is maintained.
Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset.
2. The DITHEN bit must be reset.
3. The CCIF flags must be cleared.
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 460. Data format and register coding in dithering mode

The minimum frequency is given by the following formula:

\[
\text{Resolution} = \frac{F_{\text{pwm}}}{F_{\text{Tim}}} \Rightarrow F_{\text{pwmMin}} = F_{\text{Tim}} \cdot \frac{\text{Max Resolution}}{\text{Resolution}}
\]

Dithering mode disabled: \( F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65536} \)

Dithering mode (16-bit timer): \( F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65535 + \frac{15}{16}} \)

Dithering mode (32-bit timer): \( F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{268435454 + \frac{15}{16}} \)

Note: For 16-bit timers, the maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).
For 32-bit timers, the maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFFEF.
0xFFFFFFFFF in dithering mode (corresponds to 264435454 for the integer part and 15 for the dithered part).

As shown on the Figure 461 and Figure 462 below, the dithering mode is used to increase the PWM resolution.

Figure 461. PWM resolution vs frequency (16-bit mode)

![Figure 461](image1)

Figure 462. PWM resolution vs frequency (32-bit mode)

![Figure 462](image2)

The duty cycle and/or period changes are spread over 16 consecutive periods, as described in the Figure 463 below.
The auto-reload and compare values increments are spread following specific patterns described in the Table 406 below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.
The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to '00'). In this case, the dithering pattern is applied over 8 consecutive PWM periods, considering the up and down counting phases as shown in the Figure 464 below.

**Figure 464. Dithering effect on duty cycle in center-aligned PWM mode**

<table>
<thead>
<tr>
<th>LSB value</th>
<th>PWM period</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16</td>
</tr>
<tr>
<td>0000</td>
<td>- - - - - - - - - - - - - - - - - -</td>
</tr>
<tr>
<td>0001</td>
<td>+1 - - - - - - - - - - - - - - - - -</td>
</tr>
<tr>
<td>0010</td>
<td>+1 - - - +1 - - - +1 - - - +1 - - - -</td>
</tr>
<tr>
<td>0011</td>
<td>+1 - - - +1 - - - +1 - - - +1 - - - -</td>
</tr>
<tr>
<td>0100</td>
<td>+1 - - - +1 - - - +1 - - - +1 - - - -</td>
</tr>
<tr>
<td>0101</td>
<td>+1 - +1 - +1 - - - +1 - - - +1 - - - -</td>
</tr>
<tr>
<td>0110</td>
<td>+1 - +1 - +1 - - - +1 - - - +1 - - - -</td>
</tr>
<tr>
<td>0111</td>
<td>+1 - +1 - +1 - - - +1 - - - +1 - - - -</td>
</tr>
<tr>
<td>1000</td>
<td>+1 - +1 - +1 - - - +1 - - - +1 - - - -</td>
</tr>
<tr>
<td>1001</td>
<td>+1 +1 +1 - +1 - - - +1 - - - +1 - - - -</td>
</tr>
<tr>
<td>1010</td>
<td>+1 +1 +1 - +1 - - - +1 - - - +1 - - - -</td>
</tr>
<tr>
<td>1011</td>
<td>+1 +1 +1 - +1 +1 +1 - +1 +1 +1 - +1 - -</td>
</tr>
<tr>
<td>1100</td>
<td>+1 +1 +1 - +1 +1 +1 - +1 +1 +1 - +1 - -</td>
</tr>
<tr>
<td>1101</td>
<td>+1 +1 +1 +1 +1 +1 +1 - +1 +1 +1 - +1 - -</td>
</tr>
<tr>
<td>1110</td>
<td>+1 +1 +1 +1 +1 +1 +1 - +1 +1 +1 - +1 - -</td>
</tr>
<tr>
<td>1111</td>
<td>+1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1</td>
</tr>
</tbody>
</table>

The dithering pattern is added in center-aligned PWM mode. The specific pattern for each LSB value is shown in the Table 406 below.

**Table 406. CCR and ARR register change dithering pattern**

*Note: Table 407 below shows how the dithering pattern is added in center-aligned PWM mode.*
49.4.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- \text{tim\_oc1refc} (or \text{tim\_oc2refc}) is controlled by TIMx_CCR1 and TIMx_CCR2
- \text{tim\_oc3refc} (or \text{tim\_oc4refc}) is controlled by TIMx_CCR3 and TIMx_CCR4

Asymmetric PWM mode can be selected independently on two channels (one \text{tim\_ocx} output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

\textbf{Note:} The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an \text{tim\_oc1refc} signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the \text{tim\_oc2ref} signal on channel 2, or an \text{tim\_oc2refc} signal resulting from asymmetric PWM mode 2.
Figure 465 shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 2).

**Figure 465. Generation of 2 phase-shifted PWM signals with 50% duty cycle**

![Diagram of PWM signals]

### 39.4.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_occ output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

**Note:** The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 466 shows an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1
39.4.14 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int source depends on the OCREF clear selection feature implementation, refer to Section 39.3: TIM2/TIM3/TIM4/TIM5 implementation.

If the OCREF clear selection feature is implemented, the tim_ocref_clr_int can be selected between the tim_ocref_clr input and the tim_etrf input (tim_etrf_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register. The tim_ocref_clr input can be selected among several tim_ocref_clr[7:0] inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown in Figure 467 below.
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

For example, the tim_ocref_clr_int signal can be connected to the output of a comparator to be used for current handling. In this case, tim_etrf_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 468 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.
Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), tim_ocxref is enabled again at the next counter overflow.

39.4.15 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT<CCRx ≤ ARR (in particular, 0<CCRx),

### Figure 469. Example of One-pulse mode

For example one may want to generate a positive pulse on tim_oc1 with a length of t\textsubscript{PULSE} and after a delay of t\textsubscript{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let’s use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 on tim_ti2 by writing CC2S=01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS=00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to '110 in the TIMx_SMCR register (trigger mode).
The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The \( t_{\text{DELAY}} \) is defined by the value written in the TIMx_CCR1 register.
- The \( t_{\text{PULSE}} \) is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let’s say one want to build a waveform with a transition from ‘0 to ‘1 when a compare match occurs and a transition from ‘1 to ‘0 when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to ‘0 in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register must be low.
Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CCR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CCR1 register is set to ‘0’, so the Repetitive Mode is selected.

**Particular case: tim_ocx fast enable:**

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay \( t_{\text{DELAY}} \) min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 39.4.16 Retriggerable one-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one-pulse mode described in Section 39.4.15:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

**Note:** In Retriggerable one-pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.
39.4.17 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulse width generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on the Figure 471 below.

The Figure 472 below shows how the pulse is generated for edge-aligned and encoder operating modes.
This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bit fields in TIMx_CCMR2 register.

The pulse width is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

\[ t_{PW} = PW[7:0] \times t_{PWG} \]

where \( t_{PWG} = (2^{PWPRSC[2:0]}) \times t_{tim\_ker\_ck} \).

This gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

**Note:** If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the 1st arriving trigger is extended (because of the re-trigger), while the pulse width of the last arriving trigger is correct (as shown on the Figure 473 below).
39.4.18 Encoder interface mode

**Quadrature encoder**

To select Encoder Interface mode write SMS='0001 in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS='0010 if it is counting on tim_ti2 edges only and SMS='0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCMR register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well.

The two inputs tim_ti1 and tim_ti2 are used to interface to an incremental encoder. Refer to Table 408. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1=tim_ti1 if not filtered and not inverted, tim_ti2fp2=tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the-quadrature encoder and its content, therefore, always represents the encoder’s position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.
A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder’s differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

*Figure 474* gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= 01 (TIMx_CCMR1 register, tim tí1fp1 mapped on tim tí1)
- CC2S= 01 (TIMx_CCMR2 register, tim tí2fp2 mapped on tim tí2)
- CC1P and CC1NP = ‘0’ (TIMx_CCER register, tim tí1fp1 noninverted, tim tí1fp1=tim tí1)
- CC2P and CC2NP = ‘0’ (TIMx_CCER register, tim tí2fp2 noninverted, tim tí2fp2=tim tí2)
- SMS= 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx_CR1 register, Counter is enabled)
Figure 474. Example of counter operation in encoder interface mode

![Diagram](MSv62349V1)

**Figure 475** gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P=1).

Figure 475. Example of encoder interface mode with tim_ti1fp1 polarity inverted

![Diagram](MSv62350V1)
The Figure 476 below shows the timer counter value during a speed reversal, for various counting modes.

**Figure 476. Quadrature encoder counting modes**

The timer, when configured in Encoder Interface mode provides information on the sensor’s current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register’s bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter’s most significant bit is only accessible in write mode).

**Clock plus direction encoder mode**

In addition to the quadrature encoder mode, the timer offers support other types of encoders.

In the “clock plus direction” mode shown on Figure 477, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.
This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

**Figure 477. Direction plus clock encoder mode**

**Directional Clock encoder mode**

In the “directional clock” mode on Figure 478, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to Figure 478 below) and CCxP = 1 corresponds to low-level idle state (refer to Figure 479 below).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to Figure 478 below), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to Figure 479 below).
The Table 409 here-below details how the directional clock mode operates, for any input transition.
The counter can be reset by an Index signal coming from the encoder, indicating an absolute reference position. The Index signal must be connected to the `tim_etr_in` input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMx_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Commercially available encoders are proposed with several options for index pulse conditioning, as per the Figure 480 below:

- gated with A and B: the pulse width is 1/4 of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulse width is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulse width is up to one channel period, without any alignment to the edges

<table>
<thead>
<tr>
<th>Directional clock mode</th>
<th>SMS[3:0]</th>
<th>Level on opposite signal (tim_t1fp1 for tim_t1, tim_t2fp2 for tim_t1)</th>
<th>tim_t1fp1 signal</th>
<th>tim_t2fp2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Rising</td>
<td>Falling</td>
<td>Rising</td>
</tr>
<tr>
<td>x2 mode CCxP=0</td>
<td>1100</td>
<td>High</td>
<td>Down</td>
<td>Down</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>x2 mode CCxP=1</td>
<td>1100</td>
<td>High</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>Down</td>
<td>Down</td>
</tr>
<tr>
<td>x1 mode CCxP=0</td>
<td>1101</td>
<td>High</td>
<td>No count</td>
<td>Down</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>x1 mode CCxP=1</td>
<td>1101</td>
<td>High</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>Down</td>
<td>No count</td>
</tr>
</tbody>
</table>
The circuitry tolerates jitter on index signal, whatever the gating mode, as shown on Figure 481 below.

In ungated mode, the signal must be strictly below 2 encoder periods. If the pulse width is greater or equal to 2 encoder period, the counter is reset multiple times.

The timer supports the 3 gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (i.e. channel A and channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The Index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:
- The counter is reset during up-counting (DIR bit = 0)
- The counter is set to TIMx_ARR when down counting

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. The Figure 482 below shows at which position is the index generated, for a simplistic example (an encoder providing 4 edges per mechanical rotation).
The Figure 482 below presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is '11' (ChA=1, ChB=1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the '11' state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

**Figure 482. Index generation for IPOS[1:0] = 11**

The Figure 483 below presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is '11' (ChA=1, ChB=1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the '11' state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

**Figure 483. Counter reading with index gated on channel A (IPOS[1:0] = 11)**

The Figure 484 below presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.
The Figure 484 below shows how the ‘gated on A & B’ mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

The Figure 485 and Figure 487 detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.
Figure 486. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)
The Figure 488 below shows how the index is managed in x1 and x2 modes.
**Directional index sensitivity**

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

The Figure 489 below shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

![Diagram](MSv45774V1)

**Figure 489. Directional index sensitivity**
Special first index event management

The FIDX bit in the TIMx_ECR register allows the Index to be taken only once, as shown on the Figure 490 below. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be re-armed by writing the FIDX bit to 0 and setting it again to 1.

**Figure 490. Counter reset as function of FIDX bit setting**

Index blanking

The Index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on the Figure 491 below.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input

**Figure 491. Index blanking**
Index management in non-quadrature mode

The Figure 492 and Figure 493 below detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:
- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

Encoder error management

For encoder configurations where 2 quadrature signals are available, it is possible to detect transition errors. The reading on the 2 inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on the Figure 494. below. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR status.
A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

**Figure 494. State diagram for quadrature encoded signals**

For encoder having an Index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The Index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an Index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on Figure 495 below.

**Figure 495. Up-counting encoder error detection**

![Diagram showing the error detection process](image-url)
In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on Figure 496 below, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

**Figure 496. Down-counting encode error detection**

An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

**Functional encoder Interrupts**

The following interrupts are also available in encoder mode:

- **Direction change**: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.

- **Index event**: the Index event sets the IDXF interrupt flag in the TIMx_SR status register. An Index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the Update interrupt rate, by switching from x4 to x2 to x1 mode, as show on the Figure 497 below.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMPS bit in the TIMx_SMCR register.

- SMPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting.
- SMPS = 1: the transfer is triggered by the Index event.

Figure 497. Encoder mode change with preload transferred on update (SMPS = 0)

Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register. It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

39.4.19 Direction bit output

It is possible to output a direction signal out of the timer, on the tim_oc3 and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bit field to 1011 in the TIMx_CCMR2 register.
This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

### 39.4.20 UIF bit remapping
The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

### 39.4.21 Timer input XOR function
The TI1S bit in the TIM1xx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in Section 38.3.29: Interfacing with Hall sensors on page 1508.

### 39.4.22 Timers and external trigger synchronization
The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

**Slave mode: Reset mode**
The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the
trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

**Figure 498. Control circuit in reset mode**

![Control circuit in reset mode](MSv62361V1)

**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.

3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.
**Note:** The configuration “CCxP=CCxNP=1” (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

**Slave mode: Trigger mode**

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.
Slave mode selection preload for run-time encoder mode update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode – combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode – external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_t1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on tim_etr_in and ECE=1 to enable the external clock mode 2.

2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01 in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edge only).

3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select tim_t1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on tim_t1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.
39.4.23 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

*Figure 502* and *Figure 503* show examples of master/slave timer connections.

*Figure 502. Master/Slave timer example*
Note: The timers with one channel only (see Figure 503) do not feature a master mode. However, the tim_oc1 output signal can serve as trigger for slave timer (see TIMx internal trigger connection table in Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals). The tim_oc1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger. For instance, if the destination timer tim_ker_ck clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

Using one timer as prescaler for another timer

For example, TIM_mstr can be configured to act as a prescaler for TIM_slv. Refer to Figure 502. To do this:

1. Configure TIM_mstr in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM_mstr CR2 register, a rising edge is output on tim_trgo each time an update event is generated.
2. To connect the tim_trgo output of TIM_mstr to TIM_slv, TIM_slv must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM_slv_SMCR register (writing TS=00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIM_slv_SMCR register). This causes TIM_slv to be clocked by the rising edge of the periodic TIM_mstr trigger signal (which correspond to the TIM_mstr counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If tim_ocx is selected on TIM_mstr as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM_slv.

Using one timer to enable another timer

In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to Figure 502 for connections. TIM_slv counts on the divided internal clock only when tim_oc1ref of TIM_mstr is high. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{\text{tim_cnt\_ck}} = \frac{f_{\text{tim_ker_ck}}}{3}$).
1. Configure TIM\_mstr master mode to send its Output Compare 1 Reference (tim\_oc1ref) signal as trigger output (MMS=100 in the TIM\_mstr\_CR2 register).
2. Configure the TIM\_mstr tim\_oc1ref waveform (TIM\_mstr\_CCMR1 register).
3. Configure TIM\_slv to get the input trigger from TIM\_mstr (TS=00010 in the TIM\_slv\_SMCR register).
4. Configure TIM\_slv in gated mode (SMS=101 in TIM\_slv\_SMCR register).
5. Enable TIM\_slv by writing ‘1 in the CEN bit (TIM\_slv\_CR1 register).
6. Start TIM\_mstr by writing ‘1 in the CEN bit (TIM\_mstr\_CR1 register).

**Note:** The slave timer counter clock is not synchronized with the master timer counter clock, this mode only affects the TIM\_slv counter enable signal.

![Figure 504. Gating TIM\_slv with tim\_oc1ref of TIM\_mstr](image)

In the example in *Figure 504*, the TIM\_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM\_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIM\_EGR registers.

In the next example (refer to *Figure 505*), we synchronize TIM\_mstr and TIM\_slv. TIM\_mstr is the master and starts from 0. TIM\_slv is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM\_slv stops when TIM\_mstr is disabled by writing ‘0 to the CEN bit in the TIM\_mstr\_CR1 register:
1. Configure TIM_mstr master mode to send its Output Compare 1 Reference (tim_oc1ref) signal as trigger output (MMS=100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS=101 in TIM_slv_SMCR register).
5. Reset TIM_mstr by writing ‘1’ in UG bit (TIM_mstr_EGR register).
6. Reset TIM_slv by writing ‘1’ in UG bit (TIM_slv_EGR register).
7. Initialize TIM_slv to 0xE7 by writing ‘0xE7’ in the TIM_slv counter (TIM_slv_CNT).
8. Enable TIM_slv by writing ‘1’ in the CEN bit (TIM_slv_CR1 register).
9. Start TIM_mstr by writing ‘1’ in the CEN bit (TIM_mstr_CR1 register).
10. Stop TIM_mstr by writing ‘0’ in the CEN bit (TIM_mstr_CR1 register).

**Figure 505. Gating TIM_slv with Enable of TIM_mstr**

![Diagram](MSv62377V1)

**Using one timer to start another timer**

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to Figure 502 for connections. TIM_slv starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by TIM_mstr. When TIM_slv receives the trigger signal its CEN bit is automatically set and the counter counts until we write ‘0’ to the CEN bit in the TIM_slv_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck (ftim_ker_ck = ftim_cnt_ck / 3).

1. Configure TIM_mstr master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in trigger mode (SMS=110 in TIM_slv_SMCR register).
5. Start TIM_mstr by writing ‘1’ in the CEN bit (TIM_mstr_CR1 register).
As in the previous example, both counters can be initialized before starting counting. 
*Figure 507* shows the behavior with the same configuration as in *Figure 506* but in trigger mode instead of gated mode (SMS=110 in the TIM_slv_SMCR register).

**Starting 2 timers synchronously in response to an external trigger**

In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to *Figure 502* for connections. To ensure the counters are aligned, TIM_mstr must be configured in Master/Slave mode (slave with respect to tim_ti1, master with respect to TIM_slv):
1. Configure TIM_mstr master mode to send its Enable as trigger output (MMS=001 in the TIM_mstr_CR2 register).
2. Configure TIM_mstr slave mode to get the input trigger from tim_ti1 (TS=00100 in the TIM_mstr_SMCR register).
3. Configure TIM_mstr in trigger mode (SMS=110 in the TIM_mstr_SMCR register).
4. Configure the TIM_mstr in Master/Slave mode by writing MSM=1 (TIM_mstr_SMCR register).
5. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00000 in the TIM_slv_SMCR register).
6. Configure TIM_slv in trigger mode (SMS=110 in the TIM_slv_SMCR register).

When a rising edge occurs on tim_ti1 (TIM_mstr), both counters starts counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode insert a delay between CNT_EN and CK_PSC on TIM_mstr.

Figure 508. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

39.4.24 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.
39.4.25  DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address, i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:
00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see Section 39.5.29: TIMx DMA control register (TIMx_DCR)(x = 2 to 5) for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.

2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:
   DBL = 3 transfers, DBA = 0xE and DBSS = 1.

3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).

4. Enable TIMx

5. Enable the DMA channel

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.
Note: A null value can be written to the reserved registers.

39.4.26 TIM2/TIM3/TIM4/TIM5 DMA requests

The TIM2/TIM3/TIM4/TIM5 can generate a DMA requests, as shown in Table 410.

<table>
<thead>
<tr>
<th>DMA request signal</th>
<th>DMA request</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_upd_dma</td>
<td>Update</td>
<td>UDE</td>
</tr>
<tr>
<td>tim_cc1_dma</td>
<td>Capture/compare 1</td>
<td>CC1DE</td>
</tr>
<tr>
<td>tim_cc2_dma</td>
<td>Capture/compare 2</td>
<td>CC2DE</td>
</tr>
<tr>
<td>tim_cc3_dma</td>
<td>Capture/compare 3</td>
<td>CC3DE</td>
</tr>
<tr>
<td>tim_cc4_dma</td>
<td>Capture/compare 4</td>
<td>CC4DE</td>
</tr>
<tr>
<td>tim_trg_dma</td>
<td>Trigger</td>
<td>TDE</td>
</tr>
</tbody>
</table>

Note: Some timer’s DMA requests may not be connected to the DMA controller. Refer to the DMA section(s) for more details.

39.4.27 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continues to work normally or stops.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

39.4.28 TIM2/TIM3/TIM4/TIM5 low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The timer operation is stopped and the register content is kept. No interrupt can be generated.</td>
</tr>
<tr>
<td>Standby</td>
<td>The timer is powered-down and must be reinitialized after exiting the Standby mode.</td>
</tr>
</tbody>
</table>
39.4.29  TIM2/TIM3/TIM4/TIM5 interrupts

The TIM2/TIM3/TIM4/TIM5 can generate multiple interrupts, as shown in Table 412.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop and Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_UP</td>
<td>Update</td>
<td>UIF</td>
<td>UIE</td>
<td>write 0 in UIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_CC</td>
<td>Capture/compare 1</td>
<td>CC1IF</td>
<td>CC1IE</td>
<td>write 0 in CC1IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 2</td>
<td>CC2IF</td>
<td>CC2IE</td>
<td>write 0 in CC2IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 3</td>
<td>CC3IF</td>
<td>CC3IE</td>
<td>write 0 in CC3IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 4</td>
<td>CC4IF</td>
<td>CC4IE</td>
<td>write 0 in CC4IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_TRG</td>
<td>Trigger</td>
<td>TIF</td>
<td>TIE</td>
<td>write 0 in TIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_DIRIDX</td>
<td>Index</td>
<td>IDXF</td>
<td>IDXIE</td>
<td>write 0 in IDXF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Direction</td>
<td>DIRF</td>
<td>DIRIE</td>
<td>write 0 in DIRF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_IERR</td>
<td>Index Error</td>
<td>IERRF</td>
<td>IERRIE</td>
<td>write 0 in IERRF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_TER</td>
<td>Transition Error</td>
<td>TERRF</td>
<td>TERRIE</td>
<td>write 0 in TERRF</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
39.5 TIM2/TIM3/TIM4/TIM5 registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

39.5.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5)

Address offset: 0x000

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 DITHEN: Dithering Enable
0: Dithering disabled
1: Dithering enabled

Note: The DITTHEN bit can only be modified when CEN bit is reset.

Bit 11 UIFREMAP: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 CKD[1:0]: Clock division
This bit-field indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_etr_in, tim_tix),
00: \( tDTS = t\text{tim}_\text{ker}_{\text{ck}} \)
01: \( tDTS = 2 \times t\text{tim}_\text{ker}_{\text{ck}} \)
10: \( tDTS = 4 \times t\text{tim}_\text{ker}_{\text{ck}} \)
11: Reserved

Bit 7 ARPE: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 CMS[1:0]: Center-aligned mode selection
00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)
Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter

*Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.*

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled.
These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value
(ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is
set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled

*Note: External clock, gated mode and encoder mode can work only if the CEN bit has been
previously set by software. However trigger mode can set the CEN bit automatically by
hardware.*

CEN is cleared automatically in one-pulse mode, when an update event occurs.

### 39.5.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5)

Address offset: 0x004
Reset value: 0x0000 0000

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

---

RM0481 Rev 1 1649/3119
Bits 31:26  Reserved, must be kept at reset value.
Bits 24:8  Reserved, must be kept at reset value.

Bit 7  **TI1S**: tim_ti1 selection
0: The tim_ti1_in[15:0] multiplexer output is to tim_ti1 input
1: The tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed
   and connected to the tim_ti1 input. See also Section 38.3.29: Interfacing with Hall sensors
   on page 1508.

Bits 25, 6, 5, 4  **MMS[3:0]**: Master mode selection
These bits are used to select the information to be sent in master mode to slave timers for
synchronization (tim_trgo). The combination is as follows:
0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If
   the reset is generated by the trigger input (slave mode controller configured in reset
   mode) then the signal on tim_trgo is delayed compared to the actual reset.
0001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (tim_trgo). It is
   useful to start several timers at the same time or to control a window in which a slave
   timer is enabled. The Counter Enable signal is generated by a logic AND between
   CEN control bit and the trigger input when configured in gated mode.
   When the Counter Enable signal is controlled by the trigger input, there is a delay on
   tim_trgo, except if the master/slave mode is selected (see the MSM bit description in
   TIMx_SMCR register).
0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a
   master timer can then be used as a prescaler for a slave timer.
0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to
   be set (even if it was already high), as soon as a capture or a compare match
   occurred (tim_trgo).
0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)
0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)
0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)
0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)
1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo).
   This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011,
   1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to
   unexpected behavior.
Others: Reserved

*Note*: The clock of the slave timer or ADC must be enabled prior to receive events from the
master timer, and must not be changed on-the-fly while triggers are received from the
master timer.

Bit 3  **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bits 2:0  Reserved, must be kept at reset value.
39.5.3 TIMx slave mode control register (TIMx_SMCR)\((x = 2 \text{ to } 5)\)

Address offset: 0x008  
Reset value: 0x0000 0000

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

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

**Bit 25 SMSPS**: SMS preload source  
This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active  
0: The transfer is triggered by the Timer’s Update event  
1: The transfer is triggered by the Index event

**Bit 24 SMSPES**: SMS preload enable  
This bit selects whether the SMS[3:0] bitfield is preloaded  
0: SMS[3:0] bitfield is not preloaded  
1: SMS[3:0] preload is enabled

**Bits 23:22** Reserved, must be kept at reset value.

**Bits 19:17** Reserved, must be kept at reset value.

**Bit 15 ETP**: External trigger polarity  
This bit selects whether tim_etr_in or tim_etr_in is used for trigger operations  
0: tim_etr_in is non-inverted, active at high level or rising edge  
1: tim_etr_in is inverted, active at low level or falling edge

**Bit 14 ECE**: External clock enable  
This bit enables External clock mode 2.  
0: External clock mode 2 disabled  
1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.  

**Note**: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS=111 and TS=00111). It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).  
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

**Bits 13:12** ETPS[1:0]: External trigger prescaler  
External trigger signal tim_etrp frequency must be at most 1/4 of tim_ker_ck frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etrp_in.  
00: Prescaler OFF  
01: tim_etrp frequency divided by 2  
10: tim_etrp frequency divided by 4  
11: tim_etrp frequency divided by 8
Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample `tim_etrp` signal and the length of the digital filter applied to `tim_etrp`. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

<table>
<thead>
<tr>
<th>ETF[3:0]</th>
<th>fSAMPLING</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>fDTS</td>
<td>N=0</td>
</tr>
<tr>
<td>0001</td>
<td>ftim_ker_ck</td>
<td>N=2</td>
</tr>
<tr>
<td>0010</td>
<td>ftim_ker_ck</td>
<td>N=4</td>
</tr>
<tr>
<td>0011</td>
<td>ftim_ker_ck</td>
<td>N=8</td>
</tr>
<tr>
<td>0100</td>
<td>fDTS/2</td>
<td>N=6</td>
</tr>
<tr>
<td>0101</td>
<td>fDTS/2</td>
<td>N=8</td>
</tr>
<tr>
<td>0110</td>
<td>fDTS/4</td>
<td>N=6</td>
</tr>
<tr>
<td>0111</td>
<td>fDTS/4</td>
<td>N=8</td>
</tr>
<tr>
<td>1000</td>
<td>fDTS/8</td>
<td>N=6</td>
</tr>
<tr>
<td>1001</td>
<td>fDTS/8</td>
<td>N=8</td>
</tr>
<tr>
<td>1010</td>
<td>fDTS/16</td>
<td>N=5</td>
</tr>
<tr>
<td>1011</td>
<td>fDTS/16</td>
<td>N=6</td>
</tr>
<tr>
<td>1100</td>
<td>fDTS/16</td>
<td>N=8</td>
</tr>
<tr>
<td>1101</td>
<td>fDTS/32</td>
<td>N=5</td>
</tr>
<tr>
<td>1110</td>
<td>fDTS/32</td>
<td>N=6</td>
</tr>
<tr>
<td>1111</td>
<td>fDTS/32</td>
<td>N=8</td>
</tr>
</tbody>
</table>

Bit 7 **MSM**: Master/Slave mode

- 0: No action
- 1: The effect of an event on the trigger input (`tim_trgi`) is delayed to allow a perfect synchronization between the current timer and its slaves (through `tim_trgo`). It is useful if we want to synchronize several timers on a single external event.
Bits 21, 20, 6, 5, 4  **TS[4:0]:** Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

- **00000:** Internal trigger 0 (tim_itr0)
- **00001:** Internal trigger 1 (tim_itr1)
- **00010:** Internal trigger 2 (tim_itr2)
- **00011:** Internal trigger 3 (tim_itr3)
- **00100:** tim_ti1 edge detector (tim_ti1f_ed)
- **00101:** Filtered timer input 1 (tim_ti1fp1)
- **00110:** Filtered timer input 2 (tim_ti2fp2)
- **00111:** External trigger input (tim_etrf)
- **01000:** Internal trigger 4 (tim_itr4)
- **01001:** Internal trigger 5 (tim_itr5)
- **01010:** Internal trigger 6 (tim_itr6)
- **01011:** Internal trigger 7 (tim_itr7)
- **01100:** Internal trigger 8 (tim_itr8)
- **01101:** Internal trigger 9 (tim_itr9)
- **01110:** Internal trigger 10 (tim_itr10)
- **01111:** Internal trigger 11 (tim_itr11)
- **10000:** Internal trigger 12 (tim_itr12)
- **10001:** Internal trigger 13 (tim_itr13)
- **10010:** Internal trigger 14 (tim_itr14)
- **10011:** Internal trigger 15 (tim_itr15)
- **Others:** Reserved

See [Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals](#) for product specific implementation details.

**Note:** These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3  **OCCS:** OCREF clear selection

This bit is used to select the OCREF clear source

- **0:** tim_ocref_clr_int is connected to the tim_ocref_clr input
- **1:** tim_ocref_clr_int is connected to tim_etrf

**Note:** If the OCREF clear selection feature is not supported, this bit is reserved and forced by hardware to '0'. [Section 39.3: TIM2/TIM3/TIM4/TIM5 implementation](#)
General-purpose timers (TIM2/TIM3/TIM4/TIM5)

Bits 16, 2, 1, 0 \( \text{SMS}[3:0] \): Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to
the polarity selected on the external input (see Input Control register and Control Register
description.
0000: Slave mode disabled - if CEN = `1 then the prescaler is clocked directly by the internal
clock.
0001: Encoder mode 1 - Counter counts up/down on tim_ti1fp1 edge depending on
    tim_ti2fp2 level.
0010: Encoder mode 2 - Counter counts up/down on tim_ti2fp2 edge depending on
    tim_ti1fp1 level.
0011: Encoder mode 3 - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges
depending on the level of the other input.
0100: Reset Mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the
    counter and generates an update of the registers.
0101: Gated Mode - The counter clock is enabled when the trigger input (tim_trgi) is high.
    The counter stops (but is not reset) as soon as the trigger becomes low. Both start and
    stop of the counter are controlled.
0110: Trigger Mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not
    reset). Only the start of the counter is controlled.
0111: External Clock Mode 1 - Rising edges of the selected trigger (tim_trgi) clock the
    counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi)
    reinitializes the counter, generates an update of the registers and starts the counter.
1001: Combined gated + reset mode - The counter clock is enabled when the trigger input
    (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and
    stop of the counter are controlled.
1010: Encoder mode: Clock plus direction, x2 mode.
1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by
    CC2P.
1100: Encoder mode: Directional Clock, x2 mode.
1101: Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity
    is set by CC1P and CC2P.
1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge
    sensitivity is set by CC1P.
1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge
    sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input
    (TS=00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f,
    whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo signal must be
    enabled prior to receive events from the master timer, and the clock frequency
    (prescaler) must not be changed on-the-fly while triggers are received from the master
    timer.
### 39.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5)

Address offset: 0x00C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Bit Name</th>
<th>Description</th>
<th>Reset Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-24</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>TERRIE</td>
<td>Transition error interrupt enable</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>IERRIE</td>
<td>Index error interrupt enable</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>DIRIE</td>
<td>Direction change interrupt enable</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>IDXIE</td>
<td>Index interrupt enable</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>19-15</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td>TDE</td>
<td>Trigger DMA request enable</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td>CC4DE</td>
<td>Capture/Compare 4 DMA request enable</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td>CC3DE</td>
<td>Capture/Compare 3 DMA request enable</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td>CC2DE</td>
<td>Capture/Compare 2 DMA request enable</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td>CC1DE</td>
<td>Capture/Compare 1 DMA request enable</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td>UDE</td>
<td>Update DMA request enable</td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

- **Bits 31:24** Reserved, must be kept at reset value.
- **Bit 23** TERRIE: Transition error interrupt enable  
  0: Transition error interrupt disabled  
  1: Transition error interrupt enabled
- **Bit 22** IERRIE: Index error interrupt enable  
  0: Index error interrupt disabled  
  1: Index error interrupt enabled
- **Bit 21** DIRIE: Direction change interrupt enable  
  0: Direction change interrupt disabled  
  1: Direction change interrupt enabled
- **Bit 20** IDXIE: Index interrupt enable  
  0: Index interrupt disabled  
  1: Index interrupt enabled
- **Bits 19:15** Reserved, must be kept at reset value.
- **Bit 14** TDE: Trigger DMA request enable  
  0: Trigger DMA request disabled.  
  1: Trigger DMA request enabled.
- **Bit 13** Reserved, must be kept at reset value.
- **Bit 12** CC4DE: Capture/Compare 4 DMA request enable  
  0: CC4 DMA request disabled.  
  1: CC4 DMA request enabled.
- **Bit 11** CC3DE: Capture/Compare 3 DMA request enable  
  0: CC3 DMA request disabled.  
  1: CC3 DMA request enabled.
- **Bit 10** CC2DE: Capture/Compare 2 DMA request enable  
  0: CC2 DMA request disabled.  
  1: CC2 DMA request enabled.
- **Bit 9** CC1DE: Capture/Compare 1 DMA request enable  
  0: CC1 DMA request disabled.  
  1: CC1 DMA request enabled.
- **Bit 8** UDE: Update DMA request enable  
  0: Update DMA request disabled.  
  1: Update DMA request enabled.
39.5.5 **TIMx status register (TIMx_SR)(x = 2 to 5)**

Address offset: 0x010

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value</td>
<td>rc_w0</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>rc_w0</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>rc_w0</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>rc_w0</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>rc_w0</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>rc_w0</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>rc_w0</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>rc_w0</td>
</tr>
<tr>
<td>23</td>
<td>TERRF: Transition error interrupt flag</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>IERRF: Index error interrupt flag</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>DIRF: Directional flag</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>IDXF: Index data flag</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>rc_w0</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>rc_w0</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>rc_w0</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>rc_w0</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

**Bit 23** **TERRF**: Transition error interrupt flag

This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to ‘0’.

0: No encoder transition error has been detected.

1: An encoder transition error has been detected

**Bit 22** **IERRF**: Index error interrupt flag

This flag is set by hardware when an index error is detected. It is cleared by software by writing it to ‘0’.

0: No index error has been detected.

1: An index error has been detected
Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to ‘0’.  
0: No direction change  
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to ‘0’.  
0: No index event occurred.  
1: An index event has occurred

Bits 19:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag  
Refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag  
Refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag  
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag  
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.  
0: No overcapture has been detected.  
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag  
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.  
0: No trigger event occurred.  
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag  
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag  
Refer to CC1IF description
General-purpose timers (TIM2/TIM3/TIM4/TIM5)

39.5.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5)

Address offset: 0x014
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TG</td>
<td>CC4G</td>
<td>CC3G</td>
<td>CC2G</td>
<td>CC1G</td>
<td>UG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 6 TG: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 CC4G: Capture/compare 4 generation
Refer to CC1G description

Bit 3 CC3G: Capture/compare 3 generation
Refer to CC1G description
Bit 2  **CC2G**: Capture/compare 2 generation
   Refer to CC1G description

Bit 1  **CC1G**: Capture/compare 1 generation
   This bit is set by software in order to generate an event, it is automatically cleared by hardware.
   0: No action
   1: A capture/compare event is generated on channel 1:
      **If channel CC1 is configured as output:**
      CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
      **If channel CC1 is configured as input:**
      The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0  **UG**: Update generation
   This bit can be set by software, it is automatically cleared by hardware.
   0: No action
   1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

### 39.5.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

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

**Input capture mode**

- Bits 31:16  Reserved, must be kept at reset value.
- Bits 15:12  **IC2F[3:0]**: Input capture 2 filter
- Bits 11:10  **IC2PSC[1:0]**: Input capture 2 prescaler
General-purpose timers (TIM2/TIM3/TIM4/TIM5)

Bits 9:8 \textbf{CC2S[1:0]}: Capture/compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
- 00: CC2 channel is configured as output.
- 01: CC2 channel is configured as input, \text{tim\_ic2} is mapped on \text{tim\_ti2}.
- 10: CC2 channel is configured as input, \text{tim\_ic2} is mapped on \text{tim\_ti1}.
- 11: CC2 channel is configured as input, \text{tim\_ic2} is mapped on \text{tim\_trc}. This mode is working only if an internal trigger input is selected through TS bit (\text{TIMx\_SMCR register}).

\textbf{Note}: \text{CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx\_CCER)}.

Bits 7:4 \textbf{IC1F[3:0]}: Input capture 1 filter
This bit-field defines the frequency used to sample \text{tim\_ti1} input and the length of the digital filter applied to \text{tim\_ti1}. The digital filter is made of an event counter in which \text{N} consecutive events are needed to validate a transition on the output:
- 0000: No filter, sampling is done at \text{fDTS}
- 0001: \text{fSAMPLING}=\text{ftim\_ker\_ck}, \text{N}=2
- 0010: \text{fSAMPLING}=\text{ftim\_ker\_ck}, \text{N}=4
- 0011: \text{fSAMPLING}=\text{ftim\_ker\_ck}, \text{N}=8
- 0100: \text{fSAMPLING}=\text{fDTS}/2, \text{N}=6
- 0101: \text{fSAMPLING}=\text{fDTS}/2, \text{N}=8
- 0110: \text{fSAMPLING}=\text{fDTS}/4, \text{N}=6
- 0111: \text{fSAMPLING}=\text{fDTS}/4, \text{N}=8
- 1000: \text{fSAMPLING}=\text{fDTS}/8, \text{N}=6
- 1001: \text{fSAMPLING}=\text{fDTS}/8, \text{N}=8
- 1010: \text{fSAMPLING}=\text{fDTS}/16, \text{N}=5
- 1011: \text{fSAMPLING}=\text{fDTS}/16, \text{N}=6
- 1100: \text{fSAMPLING}=\text{fDTS}/16, \text{N}=8
- 1101: \text{fSAMPLING}=\text{fDTS}/32, \text{N}=5
- 1110: \text{fSAMPLING}=\text{fDTS}/32, \text{N}=6
- 1111: \text{fSAMPLING}=\text{fDTS}/32, \text{N}=8

Bits 3:2 \textbf{IC1PSC[1:0]}: Input capture 1 prescaler
This bit-field defines the ratio of the prescaler acting on \text{CC1} input (\text{tim\_ic1}). The prescaler is reset as soon as \text{CC1E}=0 (\text{TIMx\_CCER register}).
- 00: no prescaler, capture is done each time an edge is detected on the capture input
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0 \textbf{CC1S[1:0]}: Capture/Compare 1 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, \text{tim\_ic1} is mapped on \text{tim\_ti1}
- 10: CC1 channel is configured as input, \text{tim\_ic1} is mapped on \text{tim\_ti2}
- 11: CC1 channel is configured as input, \text{tim\_ic1} is mapped on \text{tim\_trc}. This mode is working only if an internal trigger input is selected through TS bit (\text{TIMx\_SMCR register}).

\textbf{Note}: \text{CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx\_CCER)}. 
39.5.8 TIMx capture/compare mode register 1 [alternate]
(TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018
Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Output compare mode

- Bits 31:25 Reserved, must be kept at reset value.
- Bits 23:17 Reserved, must be kept at reset value.
  - Bit 15 **OC2CE**: Output compare 2 clear enable
  - Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
    refer to OC1M description on bits 6:4
  - Bit 11 **OC2PE**: Output compare 2 preload enable
  - Bit 10 **OC2FE**: Output compare 2 fast enable
  - Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
    This bit-field defines the direction of the channel (input/output) as well as the used input.
    00: CC2 channel is configured as output
    01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
    10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
    11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
    Note: **CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER)**.
  - Bit 7 **OC1CE**: Output compare 1 clear enable
    0: tim_oc1ref is not affected by the tim_ocref_clr_int input
    1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input
Bits 16, 6:4 **OC1M[3:0]:** Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (tim_oc1ref=0) as long as TIMx_CNT> TIMx_CCR1 else active (tim_oc1ref=1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT> TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved.

1011: Reserved.

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

**Note:** In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.
39.5.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

**Input capture mode**

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler
39.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C
Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

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

**Output compare mode**

Bits 31:25 Reserved, must be kept at reset value.

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

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable
39.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)

Address offset: 0x020
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>CC4NP: Capture/Compare 4 output Polarity. Refer to CC1NP description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 14</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 13</td>
<td>CC4P: Capture/Compare 4 output Polarity. Refer to CC1P description</td>
</tr>
<tr>
<td>Bit 12</td>
<td>CC4E: Capture/Compare 4 output enable. refer to CC1E description</td>
</tr>
<tr>
<td>Bit 11</td>
<td>CC3NP: Capture/Compare 3 output Polarity. Refer to CC1NP description</td>
</tr>
<tr>
<td>Bit 10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 9</td>
<td>CC3P: Capture/Compare 3 output Polarity. Refer to CC1P description</td>
</tr>
<tr>
<td>Bit 8</td>
<td>CC3E: Capture/Compare 3 output enable. Refer to CC1E description</td>
</tr>
</tbody>
</table>
Bit 7  **CC2NP**: Capture/Compare 2 output Polarity.
       Refer to CC1NP description

Bit 6  Reserved, must be kept at reset value.

Bit 5  **CC2P**: Capture/Compare 2 output Polarity.
       refer to CC1P description

Bit 4  **CC2E**: Capture/Compare 2 output enable.
       Refer to CC1E description

Bit 3  **CC1NP**: Capture/Compare 1 output Polarity.
      
      **CC1 channel configured as output**: CC1NP must be kept cleared in this case.
      **CC1 channel configured as input**: This bit is used in conjunction with CC1P to define
      tim_ti1fp1/tim_ti2fp1 polarity. refer to CC1P description.

Bit 2  Reserved, must be kept at reset value.

Bit 1  **CC1P**: Capture/Compare 1 output Polarity.

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity
of Ti1FP1 and Ti2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to Ti1FP1 rising edge
(capture or trigger operations in reset, external clock or trigger mode).
Ti1FP1 is not inverted (trigger operation in gated mode or encoder mode).
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to Ti1FP1 falling edge
(capture or trigger operations in reset, external clock or trigger mode).
Ti1FP1 is inverted (trigger operation in gated mode or encoder mode).
CC1NP=1, CC1P=0: non-inverted/both edges. The circuit is sensitive to both Ti1FP1 rising
and falling edges (capture or trigger operations in reset, external clock or trigger mode),
Ti1FP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
CC1NP=1, CC1P=1: this configuration is reserved, it must not be used.

Bit 0  **CC1E**: Capture/Compare 1 output enable.

0: Capture mode disabled / OC1 is not active
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

<table>
<thead>
<tr>
<th>CCxE bit</th>
<th>tim_ocx output state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output disabled (not driven by the timer: Hi-Z)</td>
</tr>
<tr>
<td>1</td>
<td>Output enabled (tim_ocx = tim_ocxref + Polarity)</td>
</tr>
</tbody>
</table>

**Table 413. Output control bit for standard tim_ocx channels**

**Note:** The state of the external IO pins connected to the standard tim_ocx channels depends only
on the GPIO registers when CCxE=0.
### 39.5.12 TIMx counter (TIMx_CNT)(x = 3, 4)

- **Address offset:** 0x024
- **Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>UIFCPY: Value depends on UIFREMAP in TIMx_CR1.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Bit 31** **UIFCPY:** Value depends on UIFREMAP in TIMx_CR1.
- If UIFREMAP = 0
  - Reserved
- If UIFREMAP = 1
  - **UIFCPY:** UIF Copy
    - This bit is a read-only copy of the UIF bit of the TIMx_ISR register

**Bits 30:16** Reserved, must be kept at reset value.

**Bits 15:0** **CNT[15:0]:** Counter value
- **Non-dithering mode (DITHEN = 0):**
  - The register holds the counter value.
- **Dithering mode (DITHEN = 1):**
  - The register holds the non-dithered part in CNT[15:0]. The fractional part is not available.

### 39.5.13 TIMx counter (TIMx_CNT)(x = 2, 5)

- **Address offset:** 0x024
- **Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>UIFCPY: Value depends on UIFREMAP in TIMx_CR1.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Bit 31** **UIFCPY:** Value depends on UIFREMAP in TIMx_CR1.
- If UIFREMAP = 0
  - Reserved
- If UIFREMAP = 1
  - **UIFCPY:** UIF Copy
    - This bit is a read-only copy of the UIF bit of the TIMx_ISR register

**Bits 30:16** Reserved, must be kept at reset value.

**Bits 15:0** **CNT[15:0]:** Counter value
- **Non-dithering mode (DITHEN = 0):**
  - The register holds the counter value.
- **Dithering mode (DITHEN = 1):**
  - The register holds the non-dithered part in CNT[15:0]. The fractional part is not available.
### 39.5.14 TIMx prescaler (TIMx_PSC)(x = 2 to 5)

Address offset: 0x028  
Reset value: 0x0000

<table>
<thead>
<tr>
<th>PSC[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:0 **PSC[15:0]**: Prescaler value

- The counter clock frequency `tim_cnt_ck` is equal to `f_{tim_psc_ck} / (PSC[15:0] + 1)`.  
- PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in "reset mode").

### 39.5.15 TIMx auto-reload register (TIMx_ARR)(x = 3, 4)

Address offset: 0x02C  
Reset value: 0x0000 FFFF

<table>
<thead>
<tr>
<th>Arr[19:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arr[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

- Bit 31: **UIFCPY_CNT[31]**: Value depends on UIFREMAP in TIMx_CR1.  
  - If UIFREMAP = 0  
    - **CNT[31]**: Most significant bit of counter value  
  - If UIFREMAP = 1  
    - **UIFCPY**: UIF Copy  
      This bit is a read-only copy of the UIF bit of the TIMx_ISR register

- Bits 30:0 **CNT[30:0]**: Least significant part of counter value
  - **Non-dithering mode (DITHEN = 0)**: The register holds the counter value.
  - **Dithering mode (DITHEN = 1)**: The register holds the non-dithered part in CNT[30:0]. The fractional part is not available.
### 39.5.16 TIMx auto-reload register (TIMx_ARR)(x = 2, 5)

Address offset: 0x02C  
Reset value: 0xFFFF FFFF

<table>
<thead>
<tr>
<th>Bit 31:0</th>
<th>ARR[31:0]: Auto-reload value</th>
</tr>
</thead>
</table>

ARR is the value to be loaded in the actual auto-reload register.  
Refer to the Section 39.4.3: Time-base unit on page 1576 for more details about ARR update and behavior.  
The counter is blocked while the auto-reload value is null.  
Non-dithering mode (DITHEN = 0)  
The register holds the auto-reload value.  
Dithering mode (DITHEN = 1)  
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

### 39.5.17 TIMx capture/compare register 1 (TIMx_CCR1)(x = 3, 4)

Address offset: 0x034  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:0</th>
<th>CCR1[31:16]: Capture/compare value</th>
</tr>
</thead>
</table>

CCR1[19:16]: Capture/compare value

Refer to the Section 39.4.3: Time-base unit on page 1576 for more details about CCR1 update and behavior.
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  CCR1[19:0]: Capture/compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is
loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit
OC1PE). Else the preload value is copied in the active capture/compare 1 register when an
update event occurs.
The active capture/compare register contains the value to be compared to the counter
TIMx_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the
dithered part.

If channel CC1 is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The
TIMx_CCR1 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The CCR1[15:0] bits hold the capture value. The CCR1[19:16] bits are reserved.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR1[19:0]. The CCR1[3:0] bits are reset.

39.5.18  TIMx capture/compare register 1 (TIMx_CCR1)(x = 2, 5)

Address offset: 0x034
Reset value: 0x0000 0000

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

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

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

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
 Bits 31:0  **CCR1[31:0]**: Capture/compare 1 value

**If channel CC1 is configured as output:**
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

**Non-dithering mode (DITHEN = 0)**
The register holds the compare value.

**Dithering mode (DITHEN = 1)**
The register holds the integer part in CCR1[31:4]. The CCR1[3:0] bitfield contains the dithered part.

**If channel CC1 is configured as input:**
CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

**Non-dithering mode (DITHEN = 0)**
The register holds the capture value.

**Dithering mode (DITHEN = 1)**
The register holds the capture in CCR1[31:0]. The CCR1[3:0] bits are reset.

### 39.5.19 TIMx capture/compare register 2 (TIMx_CCR2)(x = 3, 4)

Address offset: 0x038
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit no.</th>
<th>CCR2[31:0]</th>
<th>CCR2[23:16]</th>
<th>CCR2[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CCR2[15:0]**: Capture value.
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  **CCR2[19:0]**: Capture/compare 1 value

If channel CC2 is configured as output:
- CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
- The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

  - **Non-dithering mode** (DITHEN = 0)
  - The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

  - **Dithering mode** (DITHEN = 1)
  - The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:
- CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

  - **Non-dithering mode** (DITHEN = 0)
  - The CCR2[15:0] bits hold the capture value. The CCR2[19:16] bits are reserved.

  - **Dithering mode** (DITHEN = 1)
  - The register holds the capture in CCR2[19:0]. The CCR2[3:0] bits are reset.

### 39.5.20  TIMx capture/compare register 2 (TIMx_CCR2)(x = 2, 5)

Address offset: 0x038

Reset value: 0x0000 0000
Bits 31:0 $\text{CCR2}[31:0]$: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode ($\text{DITHEN} = 0$)

The register holds the compare value.

Dithering mode ($\text{DITHEN} = 1$)

The register holds the integer part in CCR2[31:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode ($\text{DITHEN} = 0$)

The register holds the capture value.

Dithering mode ($\text{DITHEN} = 1$)

The register holds the capture in CCR2[31:0]. The CCR2[3:0] bits are reset.

39.5.21 TIMx capture/compare register 3 (TIMx_CCR3)(x = 3, 4)

Address offset: 0x03C

Reset value: 0x0000 0000

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

$\text{CCR3}[31:16]$
General-purpose timers (TIM2/TIM3/TIM4/TIM5)

Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  CCR3[19:0]: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.

Dithering mode (DITHEN = 1)
The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)
The CCR3[15:0] bits hold the capture value. The CCR3[19:16] bits are reserved.

Dithering mode (DITHEN = 1)
The register holds the capture in CCR3[19:0]. The CCR3[3:0] bits are reset.

39.5.22  TIMx capture/compare register 3 (TIMx_CCR3)(x = 2, 5)

Address offset: 0x03C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>CCR3[31:16]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CCR3[15:0]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
</table>

1674/3119  RM0481 Rev 1
Bits 31:0 **CCR3[31:0]:** Capture/compare 3 value

*If channel CC3 is configured as output:*

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.

**Non-dithering mode (DITHEN = 0)**

The register holds the compare value.

**Dithering mode (DITHEN = 1)**

The register holds the integer part in CCR3[31:4]. The CCR3[3:0] bitfield contains the dithered part.

*If channel CC3 is configured as input:*

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

**Non-dithering mode (DITHEN = 0)**

The register holds the capture value.

**Dithering mode (DITHEN = 1)**

The register holds the capture in CCR3[31:0]. The CCR3[3:0] bits are reset.

### 39.5.23 TIMx capture/compare register 4 (TIMx_CCR4)(x = 3, 4)

Address offset: 0x040

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCR4[19:16]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>CCR4[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
39.5.24 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2, 5)

Address offset: 0x040
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>CCR4[31:16]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CCR4[15:0]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
</table>

Bits 31:20  Reserved, must be kept at reset value.
Bits 19:0  CCR4[19:0]: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)
The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)
The CCR4[15:0] bits hold the capture value. The CCR4[19:16] bits are reserved.

Dithering mode (DITHEN = 1)
The register holds the capture in CCR4[19:0]. The CCR4[3:0] bits are reset.
39.5.25 TIMx timer encoder control register (TIMx_ECR)(x = 2 to 5)

Address offset: 0x058

Reset value: 0x0000 0000

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

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

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

\[ t_{\text{PWG}} = (2^{\text{PWPRSC}[2:0]}) \times t_{\text{tim.ker.ck}} \]

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

\[ t_{\text{PW}} = \text{PW}[7:0] \times t_{\text{PWG}} \]

Bits 15:8 Reserved, must be kept at reset value.
General-purpose timers (TIM2/TIM3/TIM4/TIM5) RM0481

39.5.26 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5)

Address offset: 0x05C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:28</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 27:24</td>
<td>TI4SEL[3:0]</td>
</tr>
<tr>
<td>Bits 19:16</td>
<td>rw rw rw rw</td>
</tr>
<tr>
<td>Bits 15:12</td>
<td>15 14 13 12</td>
</tr>
<tr>
<td>Bits 11:8</td>
<td>11 10 9 8</td>
</tr>
<tr>
<td>Bits 7:4</td>
<td>7 6 5 4</td>
</tr>
<tr>
<td>Bits 3:0</td>
<td>3 2 1 0</td>
</tr>
</tbody>
</table>

Note: IPOS[1] bit is not significant
39.5.27 TIMx alternate function register 1 (TIMx_AF1) (x = 2 to 5)

Address offset: 0x060
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
... 1111: tim_ti4_in15
Refer to Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals for product specific implementation.

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

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH3
0001: tim_ti3_in1
... 1111: tim_ti3_in15
Refer to Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals for product specific implementation.

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

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
... 1111: tim_ti2_in15
Refer to Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals for product specific implementation.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
... 1111: tim_ti1_in15
Refer to Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals for product specific implementation.
Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 ETRSEL[3:0]: etr_in source selection
These bits select the etr_in input source.
0000: tim_etront0: TIMx_ETR input
0001: tim_etront1
... 1111: tim_etrong15
Refer to Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

39.5.28 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 5)
Address offset: 0x064
Reset value: 0x0000 0000

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 OCRSEL[2:0]: ocref_clr source selection
These bits select the ocref_clr input source.
000: tim_ocref_clr0
001: tim_ocref_clr1
... 111: tim_ocref_clr7
Refer to Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.
### 39.5.29 TIMx DMA control register (TIMx_DCR)(x = 2 to 5)

**Address offset:** 0x3DC  
**Reset value:** 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]:** DMA burst source selection  
This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).  
0000: Reserved  
0001: Update  
0010: CC1  
0011: CC2  
0100: CC3  
0101: CC4  
0110: COM  
0111: Trigger  
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.
Bits 12:8 DBL[4:0]: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on.

So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 DBA[4:0]: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

39.5.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) × 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).
### 39.5.31 TIMx register map

TIMx registers are mapped as described in the table below.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>TIMx_CR1</td>
</tr>
<tr>
<td>0x004</td>
<td>TIMx_CR2</td>
</tr>
<tr>
<td>0x008</td>
<td>TIMx_SMCR</td>
</tr>
<tr>
<td>0x00C</td>
<td>TIMx_DIER</td>
</tr>
<tr>
<td>0x010</td>
<td>TIMx_SR</td>
</tr>
<tr>
<td>0x014</td>
<td>TIMx_EGR</td>
</tr>
<tr>
<td>0x018</td>
<td>TIMx_CCMR1</td>
</tr>
<tr>
<td>0x01C</td>
<td>TIMx_CCMR2</td>
</tr>
<tr>
<td>0x020</td>
<td>TIMx_CCER</td>
</tr>
</tbody>
</table>

#### Table 414. TIM2/TIM3/TIM4/TIM5 register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIMx_CCMR1</td>
</tr>
<tr>
<td></td>
<td>TIMx_CCMR2</td>
</tr>
<tr>
<td></td>
<td>TIMx_CCER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>TIMx_CR1</td>
</tr>
<tr>
<td>0x004</td>
<td>TIMx_CR2</td>
</tr>
<tr>
<td>0x008</td>
<td>TIMx_SMCR</td>
</tr>
<tr>
<td>0x00C</td>
<td>TIMx_DIER</td>
</tr>
<tr>
<td>0x010</td>
<td>TIMx_SR</td>
</tr>
<tr>
<td>0x014</td>
<td>TIMx_EGR</td>
</tr>
<tr>
<td>0x018</td>
<td>TIMx_CCMR1</td>
</tr>
<tr>
<td>0x01C</td>
<td>TIMx_CCMR2</td>
</tr>
<tr>
<td>0x020</td>
<td>TIMx_CCER</td>
</tr>
</tbody>
</table>

### General-purpose timers (TIM2/TIM3/TIM4/TIM5)

1684/3119 RM0481 Rev 1
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register Map</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x024</td>
<td>TIMx_CNT</td>
<td>CNT[30:16]</td>
<td>0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x028</td>
<td>TIMx_PSC</td>
<td>PSC[15:0]</td>
<td>0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x02C</td>
<td>TIMx_ARR</td>
<td>ARR[19:0]</td>
<td>0000111111111111111111111111111111111111111111111111111111111111</td>
</tr>
<tr>
<td>0x034</td>
<td>TIMx_CCR1</td>
<td>CCR1[31:20]</td>
<td>0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x038</td>
<td>TIMx_CCR2</td>
<td>CCR2[31:20]</td>
<td>0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x03C</td>
<td>TIMx_CCR3</td>
<td>CCR3[31:20]</td>
<td>0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x040</td>
<td>TIMx_CCR4</td>
<td>CCR4[31:20]</td>
<td>0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x058</td>
<td>TIMx_ECR</td>
<td>PWPRSC[2:0]</td>
<td>0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x05C</td>
<td>TIMx_TISEL</td>
<td>TI4SEL[3:0]</td>
<td>0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x060</td>
<td>TIMx_AF1</td>
<td>ETRSEL[3:0]</td>
<td>0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x064</td>
<td>TIMx_AF2</td>
<td>OCRSEL[2:0]</td>
<td>0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td>0x03C</td>
<td>TIMx_DCR</td>
<td>DBSS[3:0]</td>
<td>0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBL[4:0]</td>
<td>0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DBA[4:0]</td>
<td>0000000000000000000000000000000000000000000000000000000000000000</td>
</tr>
</tbody>
</table>

Table 414. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)
Table 414. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xE0   | TIMx_DMAR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3 on page 111 for the register boundary addresses.
40 Basic timers (TIM6/TIM7)

40.1 TIM6/TIM7 introduction

The basic timers TIM6/TIM7 consist in a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time-base generation.

The basic timer can also be used for triggering the digital-to-analog converter. This is done with the trigger output of the timer.

The timers are completely independent, and do not share any resources.

40.2 TIM6/TIM7 main features

Basic timer (TIM6/TIM7) features include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow
40.3 TIM6/TIM7 functional description

40.3.1 TIM6/TIM7 block diagram

Figure 509. Basic timer block diagram

40.3.2 TIM6/TIM7 internal signals

The table in this section summarizes the TIM inputs and outputs.

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_pclk</td>
<td>Input</td>
<td>Timer APB clock</td>
</tr>
<tr>
<td>tim_ker_ck</td>
<td>Input</td>
<td>Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer: 1, 2, 3, ..., 16 (maximum value)</td>
</tr>
<tr>
<td>tim_trgo</td>
<td>Output</td>
<td>Internal trigger output. This trigger can trigger other on-chip peripherals (DAC).</td>
</tr>
<tr>
<td>tim_upd_it</td>
<td>Output</td>
<td>Timer update event interrupt</td>
</tr>
<tr>
<td>tim_upd_dma</td>
<td>Output</td>
<td>Timer update dma request</td>
</tr>
</tbody>
</table>
40.3.3 **TIM6/TIM7 clocks**

The timer bus interface is clocked by the *tim_pclk* APB clock.

The counter clock *tim_ker_ck* is connected to the *tim_pclk* input.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock *tim_ker_ck*.

*Figure 510* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

![Figure 510. Control circuit in normal mode, internal clock divided by 1](MSv62317v2)

40.3.4 **Time-base unit**

The main block of the programmable timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:
- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.
The counter is clocked by the prescaler output `tim_cnt_ck`, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal `tim_cnt_en` is set 1 clock cycle after CEN bit set.

**Prescaler description**

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 511* and *Figure 512* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

**Figure 511. Counter timing diagram with prescaler division change from 1 to 2**
40.3.5 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.
**Figure 513. Counter timing diagram, internal clock divided by 1**

- `tim_psc_clk`
- `CEN`
- `tim_cnt_clk`

Counter register:

```
31 32 33 34 35 36 00 01 02 03 04 05 06 07
```

- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)

**Figure 514. Counter timing diagram, internal clock divided by 2**

- `tim_psc_clk`
- `CEN`
- `tim_cnt_clk`

Counter register:

```
0034 0035 0036 0000 0001 0002 0003
```

- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)
Figure 515. Counter timing diagram, internal clock divided by 4

Figure 516. Counter timing diagram, internal clock divided by N
Figure 517. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)
Dithering mode

The time base effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This affects the way the TIMx_ARR is behaving, and is useful for adjusting the average counter period when the timer is used as a trigger (typically for a DAC).

The operating principle is to have the actual ARR value slightly changed (adding or not one timer clock period) over 16 consecutive counting periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average counting period.

The Figure 519 below presents the dithering principle applied to 4 consecutive counting periods.
When the dithering mode is enabled, the register coding is changed as following (see Figure 520 for example):

- the 4 LSBs are coding for the enhanced resolution part (fractional part)
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value

**Note:** The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CEN bit can be set (eventually with ARPE = 1).

**Figure 520. Data format and register coding in dithering mode**

The minimum frequency is given by the following formula:
The maximum TIMx_ARR value is limited to 0xFFFFE in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

As shown on the Figure 521 below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

**Figure 521. F_Cnt resolution vs frequency**

The period changes are spread over 16 consecutive periods, as described in the Figure 522 below.

**Figure 522. PWM dithering pattern**

The auto-reload and compare values increments are spread following specific patterns described in the Table 416 below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.
Table 416. TIMx_ARR register change dithering pattern

<table>
<thead>
<tr>
<th>LSB value</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0001</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0010</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0011</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0100</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0101</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0110</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0111</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1000</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
</tr>
<tr>
<td>1001</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
</tr>
<tr>
<td>1010</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1011</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1100</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
</tr>
<tr>
<td>1101</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>1110</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>1111</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
</tbody>
</table>

### 40.3.6 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register’s bit 31 (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.
40.3.7  **ADC triggers**

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

*Note:* The clock of the slave peripherals (such as timer, ADC) receiving the TIM_TRGO signal must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

40.3.8  **TIM6/TIM7 DMA requests**

The TIM6/TIM7 can generate a single DMA request, as shown in *Table 417.*

<table>
<thead>
<tr>
<th>DMA acronym</th>
<th>DMA request</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_upd_dma</td>
<td>Update</td>
<td>UDE</td>
</tr>
</tbody>
</table>

40.3.9  **Debug mode**

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or be stopped.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

40.3.10 **TIM6/TIM7 low-power modes**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The timer operation is stopped and the register content is kept. No interrupt can be generated.</td>
</tr>
<tr>
<td>Standby</td>
<td>The timer is powered-down and must be reinitialized after exiting the Standby mode.</td>
</tr>
</tbody>
</table>

40.3.11 **TIM6/TIM7 interrupts**

The TIM6/TIM7 can generate a single interrupt, as shown in *Table 419.*

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop and Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM6</td>
<td>Update</td>
<td>UIF</td>
<td>UIE</td>
<td>write 0 in UIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM7</td>
<td>Update</td>
<td>UIF</td>
<td>UIE</td>
<td>write 0 in UIF</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
### 40.4 TIM6/TIM7 registers

Refer to *Section 1.2 on page 101* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

#### 40.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7)

Address offset: 0x00

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:13</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 12</td>
<td><strong>DITHEN</strong>: Dithering enable</td>
</tr>
<tr>
<td>0</td>
<td>Dithering disabled</td>
</tr>
<tr>
<td>1</td>
<td>Dithering enabled</td>
</tr>
<tr>
<td>Note: The DITHEN bit can only be modified when CEN bit is reset.</td>
<td></td>
</tr>
<tr>
<td>Bit 11</td>
<td><strong>UIFREMAP</strong>: UIF status bit remapping</td>
</tr>
<tr>
<td>0</td>
<td>No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.</td>
</tr>
<tr>
<td>1</td>
<td>Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.</td>
</tr>
<tr>
<td>Bits 10:8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 7</td>
<td><strong>ARPE</strong>: Auto-reload preload enable</td>
</tr>
<tr>
<td>0</td>
<td>TIMx_ARR register is not buffered.</td>
</tr>
<tr>
<td>1</td>
<td>TIMx_ARR register is buffered.</td>
</tr>
<tr>
<td>Bits 6:4</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 3</td>
<td><strong>OPM</strong>: One-pulse mode</td>
</tr>
<tr>
<td>0</td>
<td>Counter is not stopped at update event</td>
</tr>
<tr>
<td>1</td>
<td>Counter stops counting at the next update event (clearing the CEN bit).</td>
</tr>
</tbody>
</table>
Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generates an update interrupt or DMA request if enabled.
These events can be:
  – Counter overflow/underflow
  – Setting the UG bit
  – Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
  – Counter overflow/underflow
  – Setting the UG bit
  – Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
**CEN** is cleared automatically in one-pulse mode, when an update event occurs.
### 40.4.2 TIMx control register 2 (TIMx_CR2)\((x = 6 \text{ to } 7)\)

Address offset: 0x04  
Reset value: 0x0000

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

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

Bits 6:4  **MMS[2:0]: Master mode selection**  
These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

- 000: **Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (tim_trgo).  
- 001: **Enable** - the Counter enable signal, tim_cnt_en, is used as a trigger output (tim_trgo).  
- 010: **Update** - The update event is selected as a trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

*Note:* The clock of the slave timer or the peripheral receiving the tim_trgo must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0  Reserved, must be kept at reset value.

### 40.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)\((x = 6 \text{ to } 7)\)

Address offset: 0x0C  
Reset value: 0x0000

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

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

Bit 8  **UDE**: Update DMA request enable  
0: Update DMA request disabled.  
1: Update DMA request enabled.

Bits 7:1  Reserved, must be kept at reset value.

Bit 0  **UIE**: Update interrupt enable  
0: Update interrupt disabled.  
1: Update interrupt enabled.
40.4.4 TIMx status register (TIMx_SR)(x = 6 to 7)

Address offset: 0x10
Reset value: 0x0000

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

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– On counter overflow if UDIS = 0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

40.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7)

Address offset: 0x14
Reset value: 0x0000

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

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

40.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7)

Address offset: 0x24
Reset value: 0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|
| CNT[15:0] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |


40.4.7  TIMx prescaler (TIMx_PSC)(x = 6 to 7)
Address offset: 0x28
Reset value: 0x0000

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

Bits 15:0  **PSC[15:0]**: Prescaler value

The counter clock frequency $f_{tim\_cnt\_ck}$ is equal to $f_{tim\_psc\_ck} / (PSC[15:0] + 1)$. PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register.)

40.4.8  TIMx auto-reload register (TIMx_ARR)(x = 6 to 7)
Address offset: 0x2C
Reset value: 0x0000 FFFF

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

Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register. Refer to Section 40.3.4: Time-base unit on page 1689 for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null. **Non-dithering mode (DITHEN = 0)**

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reserved.

**Dithering mode (DITHEN = 1)**

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

Bit 31  **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16  Reserved, must be kept at reset value.

Bits 15:0  **CNT[15:0]**: Counter value

**Non-dithering mode (DITHEN = 0)**

The register holds the counter value.

**Dithering mode (DITHEN = 1)**

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.
### TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

| Offset | Register name | S31 | S30 | S29 | S28 | S27 | S26 | S25 | S24 | S23 | S22 | S21 | S20 | S19 | S18 | S17 | S16 | S15 | S14 | S13 | S12 | S11 | S10 | S9 | S8 | S7 | S6 | S5 | S4 | S3 | S2 | S1 | S0 |
|--------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x00   | TIMx_CR1      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x04   | TIMx_CR2      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x08   | Reserved      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x0C   | TIMx_DIER     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x10   | TIMx_SR       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x14   | TIMx_EGR      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x18-  | Reserved      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x20   |              |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x24   | TIMx_CNT      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x28   | TIMx_PSC      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x2C   | TIMx_ARR      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

Refer to Section 2.3 on page 111 for the register boundary addresses.
41 General-purpose timers (TIM12/TIM13/TIM14)

41.1 TIM12/TIM13/TIM14 introduction

The TIM12/TIM13/TIM14 general-purpose timers consist in a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM12/TIM13/TIM14 timers are completely independent, and do not share any resources. They can be synchronized together as described in Section 41.4.20: Timer synchronization (TIM12 only).

41.2 TIM12 main features

The features of the TIM12 general-purpose timer include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization or count by internal trigger)
  - Input capture
  - Output compare
41.3 TIM13/TIM14 main features

The features of general-purpose timers TIM13/TIM14 include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed "on the fly")
- independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare
41.4 TIM12/TIM13/TIM14 functional description

41.4.1 Block diagram

Figure 523. General-purpose timer block diagram (TIM12)

Notes:
- Reg: Preload registers transferred to active registers on U event according to control bit
- Event
- Interrupt

Preload registers transferred to active registers on U event according to control bit
41.4.2 TIM12/TIM13/TIM14 pins and internal signals

Table 421 and Table 422 in this section summarize the TIM inputs and outputs.

### Table 421. TIM input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_CH1 TIM_CH2(1)</td>
<td>Input/Output</td>
<td>Timer multi-purpose channels. Each channel can be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.</td>
</tr>
</tbody>
</table>

1. Available for TIM12 only.

### Table 422. TIM internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti1_in[15:0]</td>
<td>Input</td>
<td>Internal timer inputs bus. These inputs can be used for capture or external clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_ti2_in<a href="1">15:0</a></td>
<td>Input</td>
<td>Internal trigger input bus. These inputs can be used for the slave mode controller or as an input clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_itr<a href="1">15:0</a></td>
<td>Input</td>
<td>Internal trigger input bus. These inputs can be used for the slave mode controller or as an input clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_oc1 (1)</td>
<td>Output</td>
<td>Internal timer output. Can be used for triggering other timers or the ADC(s).</td>
</tr>
</tbody>
</table>
### Table 422. TIM internal input/output signals (continued)

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_trgo(^{(1)})</td>
<td>Output</td>
<td>Internal trigger output. This trigger can trigger other on-chip peripherals.</td>
</tr>
<tr>
<td>tim_pclk</td>
<td>Input</td>
<td>Timer APB clock</td>
</tr>
<tr>
<td>tim_ker_ck</td>
<td>Input</td>
<td>Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer: 1, 2, 3, ..., 16 (maximum value)</td>
</tr>
<tr>
<td>tim_it</td>
<td>Output</td>
<td>Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests</td>
</tr>
</tbody>
</table>

1. Available for TIM12 only.
Table 423 and Table 424 list the sources connected to the `tim_ti[2:1]` input multiplexers.

**Table 423. Interconnect to the `tim_ti1` input multiplexer**

<table>
<thead>
<tr>
<th><code>tim_ti1</code> inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tim_ti1_in0</code></td>
<td>TIM12.CH1</td>
</tr>
<tr>
<td><code>tim_ti1_in1</code></td>
<td>Reserved</td>
</tr>
<tr>
<td><code>tim_ti1_in2</code></td>
<td></td>
</tr>
<tr>
<td><code>tim_ti1_in3</code></td>
<td></td>
</tr>
<tr>
<td><code>tim_ti1_in4</code></td>
<td>HSI/1024</td>
</tr>
<tr>
<td><code>tim_ti1_in5</code></td>
<td>CSI/128</td>
</tr>
<tr>
<td><code>tim_ti1_in[15:6]</code></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Table 424. Interconnect to the `tim_ti2` input multiplexer**

<table>
<thead>
<tr>
<th><code>tim_ti2</code> inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tim_ti2_in0</code></td>
<td>TIM12.CH2</td>
</tr>
<tr>
<td><code>tim_ti2_in[15:1]</code></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Table 425 lists the internal sources connected to the `tim_itr` input multiplexer.**

**Table 425. TIMx internal trigger connection**

<table>
<thead>
<tr>
<th>TIMx</th>
<th>TIM12</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tim_itr0</code></td>
<td><code>tim1_trgo</code></td>
</tr>
<tr>
<td><code>tim_itr1</code></td>
<td><code>tim2_trgo</code></td>
</tr>
<tr>
<td><code>tim_itr2</code></td>
<td><code>tim3_trgo</code></td>
</tr>
<tr>
<td><code>tim_itr3</code></td>
<td><code>tim4_trgo</code></td>
</tr>
<tr>
<td><code>tim_itr4</code></td>
<td><code>tim5_trgo</code></td>
</tr>
<tr>
<td><code>tim_itr5</code></td>
<td><code>tim8_trgo</code></td>
</tr>
<tr>
<td><code>tim_itr6</code></td>
<td>Reserved</td>
</tr>
<tr>
<td><code>tim_itr7</code></td>
<td><code>tim13_oc1</code></td>
</tr>
<tr>
<td><code>tim_itr8</code></td>
<td><code>tim14_oc1</code></td>
</tr>
<tr>
<td><code>tim_itr9</code></td>
<td><code>tim15_trgo</code></td>
</tr>
<tr>
<td><code>tim_itr10</code></td>
<td><code>tim16_oc1</code></td>
</tr>
<tr>
<td><code>tim_itr11</code></td>
<td><code>tim17_oc1</code></td>
</tr>
<tr>
<td><code>tim_itr[15:12]</code></td>
<td>Reserved</td>
</tr>
</tbody>
</table>
41.4.3 Time-base unit

The main block of the timer is a 16-bit up-counter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 525 and Figure 526 give some examples of the counter behavior when the prescaler ratio is changed on the fly.
Figure 525. Counter timing diagram with prescaler division change from 1 to 2

Figure 526. Counter timing diagram with prescaler division change from 1 to 4
41.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller on TIM12) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 527. Counter timing diagram, internal clock divided by 1
Figure 528. Counter timing diagram, internal clock divided by 2

Figure 529. Counter timing diagram, internal clock divided by 4
Figure 530. Counter timing diagram, internal clock divided by N

Figure 531. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)
Figure 532. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)

41.4.5 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1 (for TIM12): external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (for TIM12): connecting the trigger output from another timer. For instance, another timer can be configured as a prescaler for TIM12. Refer to Section : Using one timer as prescaler for another timer for more details.

Internal clock source (tim_ker_ck)

The internal clock source is the default clock source for TIM13/TIM14.

For TIM12, the internal clock source is selected when the slave mode controller is disabled (SMS='000'). The CEN bit in the TIMx_CR1 register and the UG bit in the TIMx_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 533 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.
External clock source mode 1

This mode is selected when SMS='111' in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:
1. Select the proper `tim_t[i]2_in[15:0]` source (internal or external) with the `TI2SEL[3:0]` bits in the `TIMx_TISEL` register.
2. Configure channel 2 to detect rising edges on the `tim_t[i]2` input by writing `CC2S = '01'` in the `TIMx_CCMR1` register.
3. Configure the input filter duration by writing the `IC2F[3:0]` bits in the `TIMx_CCMR1` register (if no filter is needed, keep `IC2F='0000'`).
4. Select the rising edge polarity by writing `CC2P='0'` and `CC2NP='0'` in the `TIMx_CCER` register.
5. Configure the timer in external clock mode 1 by writing `SMS='111'` in the `TIMx_SMCR` register.
6. Select `tim_t[i]2` as the trigger input source by writing `TS='110'` in the `TIMx_SMCR` register.
7. Enable the counter by writing `CEN='1'` in the `TIMx_CR1` register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on `tim_t[i]2`, the counter counts once and the `TIF` flag is set.

The delay between the rising edge on `tim_t[i]2` and the actual clock of the counter is due to the resynchronization circuit on `tim_t[i]2` input.

![Figure 535. Control circuit in external clock mode 1](image)

41.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

*Figure 536, Figure 537, Figure 538* and *Figure 539* give an overview of a capture/compare channel.

The input stage samples the corresponding `tim_t[i]x` input to generate a filtered signal `tim_t[i]xf`. Then, an edge detector with polarity selection generates a signal (`tim_t[i]xfpy`) which
can be used as trigger input by the slave mode controller or as the capture command. It is
prescaled before the capture register (ICxPS).

**Figure 536. Capture/compare channel 1 input stage (TIM13/TIM14)**

The output stage generates an intermediate waveform which is then used for reference:
tim_ocxref (active high). The polarity acts at the end of the chain.
The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 41.4.7 Input capture mode

In input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxF (TIMx_SR register) is set. CCxF can be
cleared by software by writing it to ‘0’ or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_t1 input rises. To do this, use the following procedure:

1. Select the proper tim_t1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_t1 input, so write the CC1S bits to ‘01’ in the TIMx_CCMR1 register. As soon as CC1S becomes different from ‘00’, the channel is configured in input mode and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (by programming the ICxF bits in the TIMx_CCMRx register if the input is one of the tim_tix inputs). Let’s imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on tim_t1 when 8 consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to ‘0011’ in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_t1 channel by programming CC1P and CC1NP bits to ‘00’ in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

**Note:** IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

### 41.4.8 PWM input mode (TIM12 only)

This mode allows to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIM_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges)
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These 2 tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to ‘01’ in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): program the CC1P and CC1NP bits to ‘00’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to ‘10’ in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): program the CC2P and CC2NP bits to ‘10’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to ‘00101’ in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to ‘100’ in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

**Figure 540. PWM input mode timing**

<table>
<thead>
<tr>
<th></th>
<th>tim_ti1</th>
<th>TIMx_CNT</th>
<th>TIMx_CCR1</th>
<th>TIMx_CCR2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0004</td>
<td></td>
<td>0000</td>
<td>0001</td>
<td>0002</td>
</tr>
<tr>
<td>0003</td>
<td></td>
<td></td>
<td>0004</td>
<td>0000</td>
</tr>
<tr>
<td>IC1 capture</td>
<td>IC2 capture pulse width measurement</td>
<td>IC2 capture pulse width measurement</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**41.4.9 Forced output mode**

In output mode (CCxS bits = ‘00’ in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.
To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write ‘0101’ in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP='0' (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to ‘0100’ in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

### 41.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM='0000'), be set active (OCxM='0001'), be set inactive (OCxM='0010') or can toggle (OCxM='0011') on match.
2. Sets a flag in the interrupt status register (CCxiF bit in the TIMx_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

**Procedure:**

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxiE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = ‘0011’ to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = ‘0’ to disable preload register
   - Write CCxP = ‘0’ to select active high polarity
   - Write CCxE = ‘1’ to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=’0’, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 541.
41.4.11 PWM mode

Pulse Width Modulation mode allows to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

The tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. The tim_ocx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CNT ≤ TIMx_CCRx.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then tim_ocxref is held at ‘1’. If the compare value is 0 then tim_ocxref is held at ‘0’. Figure 542 shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.
Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The Figure 543 below presents the dithering principle applied to 4 consecutive PWM cycles.
When the dithering mode is enabled, the register coding is changed as following (see Figure 544 for example):

- the 4 LSBs are coding for the enhanced resolution part (fractional part)
- the MSBs are left-shifted to the bits 19:4 and are coding for the base value.

**Note:** The following sequence must be followed when resetting the DITHEN bit:

4. CEN and ARPE bits must be reset
5. The DITHEN bit must be reset
6. The CCIF flags must be cleared
7. The CEN bit can be set (eventually with ARPE = 1).

**Figure 544. Data format and register coding in dithering mode**

The minimum frequency is given by the following formula:

\[
\text{Resolution} = \frac{F_{\text{Tim}}}{F_{\text{pwm}}} \geq F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{\text{MaxResolution}}
\]
Dithering mode disabled: \( F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65536} \)

Dithering mode enabled: \( F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{\frac{65535}{16} + \frac{15}{16}} \)

**Note:** The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

As shown on the Figure 545 below, the dithering mode allows to increase the PWM resolution whatever the PWM frequency.

**Figure 545. PWM resolution vs frequency**

The duty cycle and/or period changes are spread over 16 consecutive periods, as described in the Figure 546 below.
Figure 546. PWM dithering pattern

![Figure showing PWM dithering pattern]

The auto-reload and compare values increments are spread following specific patterns described in the **Table 426** below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

<table>
<thead>
<tr>
<th>Table 426. CCR and ARR register change dithering pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>-</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>0000</td>
</tr>
<tr>
<td>0001</td>
</tr>
<tr>
<td>0010</td>
</tr>
<tr>
<td>0011</td>
</tr>
<tr>
<td>0100</td>
</tr>
<tr>
<td>0101</td>
</tr>
<tr>
<td>0110</td>
</tr>
<tr>
<td>0111</td>
</tr>
<tr>
<td>1000</td>
</tr>
<tr>
<td>1001</td>
</tr>
<tr>
<td>1010</td>
</tr>
<tr>
<td>1011</td>
</tr>
<tr>
<td>1100</td>
</tr>
<tr>
<td>1101</td>
</tr>
</tbody>
</table>
41.4.12 Combined PWM mode (TIM12 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 547 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2
- Channel 2 is configured in PWM mode 1
41.4.13 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

\[ \text{CNT} < \text{CCR}x \leq \text{ARR} \] (in particular, \( 0 < \text{CCR}x \))
For example one may want to generate a positive pulse on tim_oc1 with a length of tPULSE and after a delay of tDELAY as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS='00110' in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).
- The tDELAY is defined by the value written in the TIMx_CCR1 register.
- The tPULSE is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let’s say one want to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M='0111' in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to ‘0’ in this example.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over...
from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

**Particular case: tim_ocx fast enable**

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay $t_{\text{DELAY min}}$ we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 41.4.14 Retriggerable one pulse mode (TIM12 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one pulse mode described in Section 41.4.13: One-pulse mode:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for retrigerrable OPM mode 1 or 2.

If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode, CCRx must be above or equal to ARR.

**Note:** The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones. This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

**Figure 549. Retriggerable one pulse mode**
41.4.15 **UIF bit remapping**

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

41.4.16 **Timer input XOR function**

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in **Figure 550**.

![Figure 550. Measuring time interval between edges on 2 signals](image)

41.4.17 **TIM12 external trigger synchronization**

The TIM12 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset mode.

**Slave mode: Reset mode**

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F='0000'). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = '01' in the TIMx_CCMR1
configure the timer in reset mode by writing SMS='100' in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS='00101' in TIMx_SMCR register.
3. Start the counter by writing CEN='1' in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

**Figure 551. Control circuit in reset mode**

---

**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F='0000'). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S='01' in TIMx_CCMR1 register. Program CC1P='1' and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS='101' in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS='00101' in TIMx_SMCR register.
3. Enable the counter by writing CEN='1' in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN='0', whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.
Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F='0000'). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S='01' in TIMx_CCMR1 register. Program CC2P='1' and CC2NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS='110' in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS='001110' in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.
41.4.18 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

41.4.19 Slave mode – combined reset + gated mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
41.4.20 Timer synchronization (TIM12 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to Section 39.4.23: Timer synchronization for details.

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

41.4.21 Using timer output as trigger for other timers (TIM13/TIM14 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.
41.4.22 ADC triggers (TIM12 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

41.4.23 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to the debug section.

41.5 TIM12/TIM13/TIM14 low-power modes

Table 427. Effect of low-power modes on TIM12/TIM13/TIM14

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The timer operation is stopped and the register content is kept. No interrupt can be generated.</td>
</tr>
<tr>
<td>Standby</td>
<td>The timer is powered-down and must be reinitialized after exiting the Standby mode.</td>
</tr>
</tbody>
</table>

41.6 TIM12/TIM13/TIM14 interrupts

The TIM12/TIM13/TIM14 can generate multiple interrupts, as shown in Table 428.

Table 428. Interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop and Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM</td>
<td>Update</td>
<td>UIF</td>
<td>UIE</td>
<td>write 0 in UIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 1</td>
<td>CC1IF</td>
<td>CC1IE</td>
<td>write 0 in CC1IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM</td>
<td>Capture/compare 2(1)</td>
<td>CC2IF</td>
<td>CC2IE</td>
<td>write 0 in CC2IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Trigger(1)</td>
<td>TIF</td>
<td>TIE</td>
<td>write 0 in TIF</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

1. Available for TIM12 only.
41.7 TIM12 registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

41.7.1 TIM12 control register 1 (TIM12_CR1)

Address offset: 0x000
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits 15:13 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 12  DITHEN: Dithering enable</td>
</tr>
<tr>
<td>0: Dithering disabled</td>
</tr>
<tr>
<td>1: Dithering enabled</td>
</tr>
<tr>
<td>Note:  The DITHEN bit can only be modified when CEN bit is reset.</td>
</tr>
<tr>
<td>Bit 11  UIFREMAP: UIF status bit remapping</td>
</tr>
<tr>
<td>0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.</td>
</tr>
<tr>
<td>1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.</td>
</tr>
<tr>
<td>Bit 10  Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 9:8 CKD[1:0]: Clock division</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit-field indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),</td>
</tr>
<tr>
<td>00: tDTS = tim_ker_ck</td>
</tr>
<tr>
<td>01: tDTS = 2 x tim_ker_ck</td>
</tr>
<tr>
<td>10: tDTS = 4 x tim_ker_ck</td>
</tr>
<tr>
<td>11: Reserved</td>
</tr>
</tbody>
</table>

| Bit 7  ARPE: Auto-reload preload enable       |
|       0: TIMx_ARR register is not buffered.   |
|       1: TIMx_ARR register is buffered.       |

| Bits 6:4 Reserved, must be kept at reset value. |

| Bit 3  OPM: One-pulse mode                    |
|       0: Counter is not stopped on the update event |
|       1: Counter stops counting on the next update event (clearing the CEN bit). |
Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
- 0: Any of the following events generates an update interrupt if enabled. These events can be:
  - Counter overflow
  - Setting the UG bit
  - Update generation through the slave mode controller
- 1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable update event (UEV) generation.
- 0: UEV enabled. An UEV is generated by one of the following events:
  - Counter overflow
  - Setting the UG bit
  Buffered registers are then loaded with their preload values.
- 1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable
- 0: Counter disabled
- 1: Counter enabled
  CEN is cleared automatically in one-pulse mode, when an update event occurs.

*Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

### 41.7.2 TIM12 control register 2 (TIM12_CR2)

Address offset: 0x004
Reset value: 0x0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>T11S</th>
<th>MMS[2:0]</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

** rw rw rw rw **
41.7.3 TIM12 slave mode control register (TIM12_SMCR)

Address offset: 0x008
Reset value: 0x0000 0000

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

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

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 MSM: Master/Slave mode
0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful in order to synchronize several timers on a single external event.
Bits 21, 20, 6, 5, 4  **TS[4:0]: Trigger selection**

This TS[4:0] bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
01111: Internal Trigger 10 (tim_itr11)
10000: Internal Trigger 10 (tim_itr12)
10001: Internal Trigger 10 (tim_itr13)
10010: Internal Trigger 10 (tim_itr14)
10011: Internal Trigger 10 (tim_itr15)

Others: Reserved

See *Table 425: TIMx internal trigger connection* for more details on the meaning of tim_itrX for each timer.

*Note*: These bits must be changed only when they are not used (for example when SMS='000') to avoid wrong edge detections at the transition.

Bit 3  Reserved, must be kept at reset value.
Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.
0001: Reserved
0010: Reserved
0011: Reserved
0100: Reset Mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.
0101: Gated Mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110: Trigger Mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.
0111: External Clock Mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.
1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
Other codes: reserved.

*Note:* The gated mode (including gated + reset mode) must not be used if tim_ti1f_ed is selected as the trigger input (TS='00100'). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

*Note:* The clock of the slave peripherals (timer, ADC,...) receiving the tim_trgo signals must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

### 41.7.4 TIM12 Interrupt enable register (TIM12_DIER)

**Address offset:** 0x00C  
**Reset value:** 0x0000

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

<table>
<thead>
<tr>
<th>TIE</th>
<th>Res.</th>
<th>Res.</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

**Bit 6** **TIE**: Trigger interrupt enable

0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

**Bits 5:3** Reserved, must be kept at reset value.
Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
- 0: CC2 interrupt disabled.
- 1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
- 0: CC1 interrupt disabled.
- 1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
- 0: Update interrupt disabled.
- 1: Update interrupt enabled.

### 41.7.5 TIM12 status register (TIM12_SR)

Address offset: 0x010

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag

Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.

- 0: No overcapture has been detected.
- 1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

- 0: No trigger event occurred.
- 1: Trigger interrupt pending.

Bits 5:3 Reserved, must be kept at reset value.
41.7.6 TIM12 event generation register (TIM12_EGR)

Address offset: 0x014
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TG</td>
<td></td>
<td></td>
<td>CC2G</td>
<td>CC1G</td>
<td>UG</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action
1: The TIF flag is set in the TIMx_SR register. Related interrupt can occur if enabled

Bits 5:3 Reserved, must be kept at reset value.
41.7.7 TIM12 capture/compare mode register 1 (TIM12_CCMR1)

Address offset: 0x018
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

**Input capture mode:**

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

**Bits 31:16**  Reserved, must be kept at reset value.

**Bits 15:12**  IC2F[3:0]: Input capture 2 filter

**Bits 11:10**  IC2PSC[1:0]: Input capture 2 prescaler

**Bits 9:8**  CC2S[1:0]: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TIM_t12
10: CC2 channel is configured as input, IC2 is mapped on TIM_t1f
11: CC2 channel is configured as input, IC2 is mapped on TIM_trc. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

*Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).*
Bits 7:4  **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample the tim_t1 input and the length of the
digital filter applied to tim_t1. The digital filter is made of an event counter in which N
consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at fDTS
- 0001: \( f_{\text{SAMPLING}} = f_{\text{tim_ker_ck}} \), N=2
- 0010: \( f_{\text{SAMPLING}} = f_{\text{tim_ker_ck}} \), N=4
- 0011: \( f_{\text{SAMPLING}} = f_{\text{tim_ker_ck}} \), N=8
- 0100: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/2 \), N=6
- 0101: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/2 \), N=8
- 0110: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/4 \), N=6
- 0111: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/4 \), N=8
- 1000: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/8 \), N=6
- 1001: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/8 \), N=8
- 1010: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/16 \), N=5
- 1011: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/16 \), N=6
- 1100: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/16 \), N=8
- 1101: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/32 \), N=5
- 1110: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/32 \), N=6
- 1111: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/32 \), N=8

Bits 3:2  **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

- 00: no prescaler, capture is done each time an edge is detected on the capture input
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0  **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, tim_ic1 is mapped on tim_t1
- 10: CC1 channel is configured as input, tim_ic1 is mapped on tim_t2
- 11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working
  only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note*: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

### 41.7.8 TIM12 capture/compare mode register 1 [alternate]

**(TIM12_CCMR1)**

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture
mode (previous section). The direction of a channel is defined by configuring the
 corresponding CCxS bits. All the other bits in this register have different functions in input
and output modes.
### Output compare mode:

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23-17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24-14</td>
<td><strong>OC2M[3:0]</strong>: Output compare 2 mode</td>
</tr>
<tr>
<td></td>
<td>Refer to OC1M[3:0] for bit description.</td>
</tr>
<tr>
<td>11</td>
<td><strong>OC2PE</strong>: Output compare 2 preload enable</td>
</tr>
<tr>
<td>10</td>
<td><strong>OC2FE</strong>: Output compare 2 fast enable</td>
</tr>
<tr>
<td>9-8</td>
<td><strong>CC2S[1:0]</strong>: Capture/Compare 2 selection</td>
</tr>
<tr>
<td></td>
<td>This bitfield defines the direction of the channel (input/output) as well as the used input.</td>
</tr>
<tr>
<td></td>
<td>00: CC2 channel is configured as output.</td>
</tr>
<tr>
<td></td>
<td>01: CC2 channel is configured as input, IC2 is mapped on TIM_TI2</td>
</tr>
<tr>
<td></td>
<td>10: CC2 channel is configured as input, IC2 is mapped on TIM_TI1</td>
</tr>
<tr>
<td></td>
<td>11: CC2 channel is configured as input, IC2 is mapped on TIM_TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).</td>
</tr>
<tr>
<td>7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas the active level of tim_oc1 depends on the CC1P.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. The tim_oc1ref signal is forced high when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. The tim_oc1ref signal is forced low when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT=TIMx_CCR1

0100: Force inactive level - tim_oc1ref is forced low

0101: Force active level - tim_oc1ref is forced high

0110: PWM mode 1 - channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else it is inactive

0111: PWM mode 2 - channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else it is active

1000: Retriggable OPM mode 1 - The channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1001: Retriggable OPM mode 2 - The channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1ref is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1ref is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Reserved,

1111: Reserved

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.
Bit 3 **OC1PE**: Output compare 1 preload enable
   0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken into account immediately
   1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded into the active register at each update event

Bit 2 **OC1FE**: Output compare 1 fast enable
   This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
   0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the CC1 output when an edge occurs on the trigger input is 5 clock cycles
   1: An active edge on the trigger input acts like a compare match on the CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection
   This bitfield defines the direction of the channel (input/output) as well as the used input.
   00: CC1 channel is configured as output
   01: CC1 channel is configured as input, tim_ic1 is mapped on tim_tl1
   10: CC1 channel is configured as input, tim_ic1 is mapped on tim_tl2
   11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

   **Note**: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

### 41.7.9 TIM12 capture/compare enable register (TIM12_CCER)

**Address offset**: 0x020

**Reset value**: 0x0000

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

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity
   Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity
   Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
   Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity
   CC1 channel configured as output: CC1NP must be kept cleared
   CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define tim_tl1fp1/tim_tl2fp1 polarity (refer to CC1P description).
Bit 2 Reserved, must be kept at reset value.

Bit 1 CC1P: Capture/Compare 1 output Polarity.

0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of tim_t1fp1 and tim_t2fp1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to tim_t1fp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), tim_t1fp1 is not inverted (trigger operation in gated mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to tim_t1fp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_t1fp1 is inverted (trigger operation in gated mode).

CC1NP=1, CC1P=0: non-inverted/both edges. The circuit is sensitive to both tim_t1fp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_t1fp1 is not inverted (trigger operation in gated mode).

CC1NP=1, CC1P=1: This configuration is reserved, it must not be used.

Bit 0 CC1E: Capture/Compare 1 output enable.

0: Capture mode disabled / tim_oc1 is not active
1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

<table>
<thead>
<tr>
<th>CCxE bit</th>
<th>tim_ocx output state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output disabled (not driven by the timer: Hi-Z)</td>
</tr>
<tr>
<td>1</td>
<td>Output enabled (tim_ocx = tim_ocxref + Polarity)</td>
</tr>
</tbody>
</table>

Table 429. Output control bit for standard tim_ocx channels

Note: The states of the external I/O pins connected to the standard tim_ocx channels depend on the state of the tim_ocx channel and on the GPIO registers.

41.7.10 TIM12 counter (TIM12_CNT)

Address offset: 0x024
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>UIF</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

CNT[15:0]

1752/3119 RM0481 Rev 1
Bit 31  **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16  Reserved, must be kept at reset value.

Bits 15:0  **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)
The register holds the counter value.

Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

### 41.7.11 TIM12 prescaler (TIM12_PSC)

Address offset: 0x028  
Reset value: 0x0000

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

Bits 15:0  **PSC[15:0]**: Prescaler value

The counter clock frequency \( f_{\text{tim_cnt_ck}} \) is equal to \( f_{\text{tim_psc_ck}} / (PSC[15:0] + 1) \).
PSC contains the value to be loaded into the active prescaler register at each update event.
(including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in "reset mode").

### 41.7.12 TIM12 auto-reload register (TIM12_ARR)

Address offset: 0x02C  
Reset value: 0x0000 FFFF

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

Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to the Section 41.4.3: Time-base unit on page 1712 for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)
The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.
41.7.13 TIM12 capture/compare register 1 (TIM12_CCR1)

Address offset: 0x034
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

CCR1[15:0]

Bits 31:20 Reserved, must be kept at reset value.
Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

**If channel CC1 is configured as output:**
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.
- **Non-dithering mode (DITHEN = 0)**
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
- **Dithering mode (DITHEN = 1)**
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

**If channel CC1 is configured as input:**
CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.
- **Non-dithering mode (DITHEN = 0)**
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
- **Dithering mode (DITHEN = 1)**
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

41.7.14 TIM12 capture/compare register 2 (TIM12_CCR2)

Address offset: 0x038
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

CCR2[15:0]

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

1754/3119 RM0481 Rev 1
Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)
The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)
The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

### 41.7.15 TIM12 timer input selection register (TIM12_TISEL)

Address offset: 0x05C

Reset value: 0x0000

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input

0000: TIMx_CH2 input (tim_ti2_in0)
0001: tim_ti2_in1
...
0100: tim_ti2_in15
Refer to *Table 424: Interconnect to the tim_ti2 input multiplexer* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input

0000: TIMx_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Table 423: Interconnect to the tim_ti1 input multiplexer* for interconnects list.
## 41.7.16 TIM12 register map

TIM12 registers are mapped as 16-bit addressable registers as described below:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>TIM12_CR1</td>
<td>0x004</td>
<td>TIM12_CR2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x008</td>
<td>TIM12_SMCR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00C</td>
<td>TIM12_DIER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x010</td>
<td>TIM12_SR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x014</td>
<td>TIM12_EGR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x018</td>
<td>TIM12_CCMR1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x020</td>
<td>TIM12_CCER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x024</td>
<td>TIM12_CNT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x028</td>
<td>TIM12_PSC</td>
</tr>
</tbody>
</table>

### Table 430. TIM12 register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>TIM12_CR1</td>
<td>0x004</td>
<td>TIM12_CR2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x008</td>
<td>TIM12_SMCR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00C</td>
<td>TIM12_DIER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x010</td>
<td>TIM12_SR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x014</td>
<td>TIM12_EGR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x018</td>
<td>TIM12_CCMR1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x020</td>
<td>TIM12_CCER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x024</td>
<td>TIM12_CNT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x028</td>
<td>TIM12_PSC</td>
</tr>
</tbody>
</table>

**Reset value**

- **TIM12_CR1**: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- **TIM12_CR2**: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- **TIM12_SMCR**: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- **TIM12_DIER**: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- **TIM12_SR**: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- **TIM12_EGR**: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- **TIM12_CCMR1**: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- **TIM12_CCER**: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- **TIM12_CNT**: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- **TIM12_PSC**: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Refer to Section 2.3 on page 111 for the register boundary addresses.
41.8 TIM13/TIM14 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

41.8.1 TIMx control register 1 (TIMx_CR1)(x = 13, 14)

Address offset: 0x000
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled

*Note: The DITHEN bit can only be modified when CEN bit is reset.*

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),
00: tDTS = tim_ker_ck
01: tDTS = 2 × tim_ker_ck
10: tDTS = 4 × tim_ker_ck
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).
Bit 2  **URS:** Update request source
This bit is set and cleared by software to select the update interrupt (UEV) sources.
0: Any of the following events generate an UEV if enabled:
   – Counter overflow
   – Setting the UG bit
1: Only counter overflow generates an UEV if enabled.

Bit 1  **UDIS:** Update disable
This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.
0: UEV enabled. An UEV is generated by one of the following events:
   – Counter overflow
   – Setting the UG bit.
   Buffered registers are then loaded with their preload values.
1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0  **CEN:** Counter enable
0: Counter disabled
1: Counter enabled

*Note:* External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

### 41.8.2 TIMx Interrupt enable register (TIMx_DIER)(x = 13, 14)
Address offset: 0x00C
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 15:2  Reserved, must be kept at reset value.

Bit 1  **CC1IE:** Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0  **UIE:** Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

### 41.8.3 TIMx status register (TIMx_SR)(x = 13, 14)
Address offset: 0x010
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

|    | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 |

ST
Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set.

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred.

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on TIM_ic1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCRER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow and if UDIS=’0’ in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=’0’ and UDIS=’0’ in the TIMx_CR1 register.

### 41.8.4 TIMx event generation register (TIMx_EGR)(x = 13, 14)
Address offset: 0x014
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CC1G** **UG**

W W
Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
- **If channel CC1 is configured as output**: CC1IF flag is set, corresponding interrupt or is sent if enabled.
- **If channel CC1 is configured as input**: The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared.

### 41.8.5 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 13, 14)

Address offset: 0x018
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

#### Input capture mode:

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.
41.8.6  TIMx capture/compare mode register 1 [alternate]  
(TIMx_CCMR1)(x = 13, 14)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.
## Output compare mode:

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>OC1M[2:0]</td>
<td>OC1PE</td>
<td>OC1FE</td>
<td>CC1S[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Bits 31:17* Reserved, must be kept at reset value.

*Bits 15:7* Reserved, must be kept at reset value.

*Bits 16, 6, 4 OC1M[3:0]:* Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

0000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT < TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1 else active.

Others: Reserved

*Note:* In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.
41.8.7 **TIMx capture/compare enable register**

**(TIMx_CCER)(x = 13, 14)**

Address offset: 0x020

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>3</td>
<td>CC1NP</td>
<td>Capture/Compare 1 complementary output Polarity</td>
</tr>
<tr>
<td>2</td>
<td>CC1P</td>
<td>Capture/Compare 1 output Polarity</td>
</tr>
<tr>
<td>1</td>
<td>CC1E</td>
<td>Capture/Compare 1 enable</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Bits 15:4 Reserved, must be kept at reset value.

**Bit 3 CC1NP**: Capture/Compare 1 complementary output Polarity.

CC1 channel configured as output: CC1NP must be kept cleared.
CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define tim_ti1fp1 polarity (refer to CC1P description).
Bit 2  Reserved, must be kept at reset value.

Bit 1  **CC1P**: Capture/Compare 1 output Polarity.

0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of tim_ti1fp1 for capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to tim_ti1fp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 is not inverted (trigger operation in gated mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to tim_ti1fp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 is inverted (trigger operation in gated mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both tim_ti1fp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 not inverted (trigger operation in gated mode).

CC1NP=1, CC1P=0: This configuration is reserved, it must not be used.

Bit 0  **CC1E**: Capture/Compare 1 output enable.

0: Capture mode disabled / tim_oc1 is not active
1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

<table>
<thead>
<tr>
<th>CCxE bit</th>
<th>tim_ocx output state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output Disabled (tim_ocx='0')</td>
</tr>
<tr>
<td>1</td>
<td>tim_ocx=tim_ocxref + Polarity</td>
</tr>
</tbody>
</table>

**Note**: The state of the external I/O pins connected to the standard tim_ocx channels depends on the tim_ocx channel state and the GPIO registers.

**41.8.8 TIMx counter (TIMx_CNT)(x = 13, 14)**

Address offset: 0x024
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>CNT[15:0]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

**Table 431. Output control bit for standard tim_ocx channels**

<table>
<thead>
<tr>
<th>CCxE bit</th>
<th>tim_ocx output state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output Disabled (tim_ocx='0')</td>
</tr>
<tr>
<td>1</td>
<td>tim_ocx=tim_ocxref + Polarity</td>
</tr>
</tbody>
</table>

**Note**: The state of the external I/O pins connected to the standard tim_ocx channels depends on the tim_ocx channel state and the GPIO registers.
41.8.9 TIMx prescaler (TIMx_PSC)\((x = 13, 14)\)

Address offset: 0x028

Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:0 PSC[15:0]: Prescaler value

- The counter clock frequency \( \text{tim}_\text{cnt}_\text{ck} \) is equal to \( \frac{f_{\text{tim}_\text{psc}_\text{ck}}}{(PSC[15:0] + 1)} \).
- PSC contains the value to be loaded in the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

41.8.10 TIMx auto-reload register (TIMx_ARR)\((x = 13, 14)\)

Address offset: 0x02C

Reset value: 0x0000 FFFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 ARR[19:0]: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the Section 41.4.3: Time-base unit on page 1712 for more details about ARR update and behavior.

- The counter is blocked while the auto-reload value is null.
- Non-dithering mode (DITHEN = 0)
- The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.
- Dithering mode (DITHEN = 1)
41.8.11 TIMx capture/compare register 1 (TIMx_CCR1)(x = 13, 14)

Address offset: 0x034
Reset value: 0x0000 0000

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 CCR1[19:0]: Capture/compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register
(bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when
an update event occurs.
The active capture/compare register contains the value to be compared to the counter
TIMx_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the
dithered part.

If channel CC1 is configured as input:
CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The
TIMx_CCR1 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

41.8.12 TIMx timer input selection register (TIMx_TISEL)(x = 13, 14)

Address offset: 0x05C
Reset value: 0x0000
41.8.13 TIM13/TIM14 register map

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

### Table 432. TIM13/TIM14 register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>TIMx_CR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C</td>
<td>TIMx_DIER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>TIMx_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>TIMx_EGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x018</td>
<td>TIMx_CCMR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output compare mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01C</td>
<td>TIMx_CCMR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Input capture mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x020</td>
<td>TIMx_CCER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x024</td>
<td>TIMx_CNT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 432. TIM13/TIM14 register map and reset values (continued)

| Offset | Register name | Offset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       | Reset value | Name       |Reset to Section 2.3 on page 111 for the register boundary addresses.
42 General purpose timers (TIM15/TIM16/TIM17)

42.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in Section 42.4.26: Timer synchronization (TIM15 only).

42.2 TIM15 main features

TIM15 includes the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)
42.3 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input
42.4 TIM15/TIM16/TIM17 functional description

42.4.1 Block diagram

Figure 554. TIM15 block diagram

Notes:

- Preload registers transferred to active registers on U event according to control bit
- Event
- Interrupt & DMA output

1. Refer to Section 42.4.15: Using the break function for details.
42.4.2 TIM15/TIM16/TIM17 pins and internal signals

Table 433 and Table 434 in this section summarize the TIM inputs and outputs.

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_CH1</td>
<td>Input/Output</td>
<td>Timer multi-purpose channels. Each channel can be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.</td>
</tr>
<tr>
<td>TIM_CH1N</td>
<td>Output</td>
<td>Timer complementary outputs, derived from TIM_CH1 output with the possibility to have deadtime insertion.</td>
</tr>
<tr>
<td>TIM_BKIN</td>
<td>Input / Output</td>
<td>Break input. This input can also be configured in bidirectional mode.</td>
</tr>
</tbody>
</table>

1. Available for TIM15 only.
Table 434. TIM internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti1_in[15:0]</td>
<td>Input</td>
<td>Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_ti2_in<a href="1">15:0</a></td>
<td>Input</td>
<td>Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_itr<a href="1">15:0</a></td>
<td>Input</td>
<td>Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_trgo(1)</td>
<td>Output</td>
<td>Internal trigger output. This trigger can trigger other on-chip peripherals.</td>
</tr>
<tr>
<td>tim_ocref_clr[7:0]</td>
<td>Input</td>
<td>Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.</td>
</tr>
<tr>
<td>tim_brk_cmp[8:1]</td>
<td>Input</td>
<td>Break input for internal signals</td>
</tr>
<tr>
<td>tim_sys_brk[n:0]</td>
<td>Input</td>
<td>System break input. This input gathers the MCU's system level errors.</td>
</tr>
<tr>
<td>tim_pclk</td>
<td>Input</td>
<td>Timer APB clock</td>
</tr>
<tr>
<td>tim_ker_ck</td>
<td>Input</td>
<td>Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3, ..., 16 (maximum value)</td>
</tr>
<tr>
<td>tim_it</td>
<td>Output</td>
<td>Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests</td>
</tr>
<tr>
<td>tim_cc1_dma</td>
<td>Output</td>
<td>Timer capture / compare 1 dma request</td>
</tr>
<tr>
<td>tim_upd_dma</td>
<td>Output</td>
<td>Timer update dma request</td>
</tr>
<tr>
<td>tim_trg_dma</td>
<td>Output</td>
<td>Timer trigger dma request</td>
</tr>
<tr>
<td>tim_com_dma</td>
<td>Output</td>
<td>Timer commutation dma request</td>
</tr>
</tbody>
</table>

1. Available for TIM15 only.
Table 435 and Table 436 list the sources connected to the tim_ti[2:1] input multiplexers.

Table 435. Interconnect to the tim_ti1 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti1 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM15</td>
</tr>
<tr>
<td>tim_ti1_in0</td>
<td>TIM15_CH1</td>
</tr>
<tr>
<td>tim_ti1_in1</td>
<td>TIM2_CH1</td>
</tr>
<tr>
<td>tim_ti1_in2</td>
<td>TIM3_CH1</td>
</tr>
<tr>
<td>tim_ti1_in3</td>
<td>TIM4_CH1</td>
</tr>
<tr>
<td>tim_ti1_in4</td>
<td>RCC_LSE</td>
</tr>
<tr>
<td>tim_ti1_in5</td>
<td>RCC_CSI/128</td>
</tr>
<tr>
<td>tim_ti1_in[15:6]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 436. Interconnect to the tim_ti2 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti2 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM15</td>
</tr>
<tr>
<td>tim_ti2_in0</td>
<td>TIM15_CH2</td>
</tr>
<tr>
<td>tim_ti2_in1</td>
<td>TIM2_CH2</td>
</tr>
<tr>
<td>tim_ti2_in2</td>
<td>TIM3_CH2</td>
</tr>
<tr>
<td>tim_ti2_in3</td>
<td>TIM4_CH2</td>
</tr>
<tr>
<td>tim_ti2_in[15:4]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 437 lists the internal sources connected to the tim_itr input multiplexer.

Table 437. TIMx internal trigger connection

<table>
<thead>
<tr>
<th>tim_itrx inputs</th>
<th>TIM15</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_itr0</td>
<td>tim1_trgo</td>
</tr>
<tr>
<td>tim_itr1</td>
<td>tim2_trgo</td>
</tr>
<tr>
<td>tim_itr2</td>
<td>tim3_trgo</td>
</tr>
<tr>
<td>tim_itr3</td>
<td>tim4_trgo</td>
</tr>
<tr>
<td>tim_itr4</td>
<td>tim5_trgo</td>
</tr>
<tr>
<td>tim_itr5</td>
<td>tim8_trgo</td>
</tr>
<tr>
<td>tim_itr6</td>
<td>tim12_trgo</td>
</tr>
<tr>
<td>tim_itr7</td>
<td>tim13_oc1</td>
</tr>
<tr>
<td>tim_itr8</td>
<td>tim14_oc1</td>
</tr>
<tr>
<td>tim_itr9</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
42.4.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:
- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

Table 437. TIMx internal trigger connection (continued)

<table>
<thead>
<tr>
<th>tim_itr inputs</th>
<th>TIM15</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_itr10</td>
<td>tim16_oc1</td>
</tr>
<tr>
<td>tim_itr11</td>
<td>tim17_oc1</td>
</tr>
<tr>
<td>tim_itr[15:12]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 438 and Table 439 list the sources connected to the tim_brk input.

Table 438. Timer break interconnect

<table>
<thead>
<tr>
<th>tim_brk inputs</th>
<th>TIM15</th>
<th>TIM16</th>
<th>TIM17</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_BKIN</td>
<td>TIM15_BKIN pin</td>
<td>TIM16_BKIN pin</td>
<td>TIM17_BKIN pin</td>
</tr>
<tr>
<td>tim_brk_cmp[8:1]</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 439. System break interconnect

<table>
<thead>
<tr>
<th>tim_sys_brk inputs</th>
<th>TIM15/16/17</th>
<th>Enable bit in SBS_CFRG2 register</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_sys_brk0</td>
<td>FLASH double ECC error</td>
<td>ECCL</td>
</tr>
<tr>
<td>tim_sys_brk1</td>
<td>Programmable Voltage Detector (PVD)</td>
<td>PVDL</td>
</tr>
<tr>
<td>tim_sys_brk2</td>
<td>SRAM double ECC error</td>
<td>SEL</td>
</tr>
<tr>
<td>tim_sys_brk3</td>
<td>Cortex®-M33 LOCKUP</td>
<td>CLL</td>
</tr>
<tr>
<td>CSS</td>
<td>Clock Security System</td>
<td>None (always enabled)</td>
</tr>
</tbody>
</table>
The counter is clocked by the prescaler output `tim_cnt_ck`, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

**Prescaler description**

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 556* and *Figure 557* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

**Figure 556. Counter timing diagram with prescaler division change from 1 to 2**

![Counter timing diagram](image)
42.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.
When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

**Figure 558. Counter timing diagram, internal clock divided by 1**
Figure 559. Counter timing diagram, internal clock divided by 2

Figure 560. Counter timing diagram, internal clock divided by 4
Figure 561. Counter timing diagram, internal clock divided by N

Figure 562. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)
### 42.4.5 Repetition counter

**Section 42.4.3: Time-base unit** describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to Figure 564). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

![Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)](image)
42.4.6 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to Using one timer to enable another timer for more details.

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS=000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed.
only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock \( \text{tim\_ker\_ck} \).

*Figure 565* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

*Figure 565. Control circuit in normal mode, internal clock divided by 1*

![Control circuit in normal mode, internal clock divided by 1](image)

**External clock source mode 1**

This mode is selected when \( \text{SMS}=111 \) in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

*Figure 566. \text{tim\_ti2} external clock connection example*

![tim\_ti2 external clock connection example](image)

For example, to configure the upcounter to count in response to a rising edge on the \text{tim\_ti2} input, use the following procedure:
1. Select the proper `tim_ti2_in[15:0]` source (internal or external) with the `TI2SEL[3:0]` bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the `tim_ti2` input by writing `CC2S = '01'` in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the `IC2F[3:0]` bits in the TIMx_CCMR1 register (if no filter is needed, keep `IC2F=0000`).
4. Select rising edge polarity by writing `CC2P=0` in the TIMx_CCRER register.
5. Configure the timer in external clock mode 1 by writing `SMS=111` in the TIMx_SMCR register.
6. Select `tim_ti2` as the trigger input source by writing `TS=00110` in the TIMx_SMCR register.
7. Enable the counter by writing `CEN=1` in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on `tim_ti2`, the counter counts once and the `TIF` flag is set.

The delay between the rising edge on `tim_ti2` and the actual clock of the counter is due to the resynchronization circuit on `tim_ti2` input.

**Figure 567. Control circuit in external clock mode 1**

<table>
<thead>
<tr>
<th>tim_ti2</th>
<th>CEN</th>
<th><code>tim_cnt_ck, tim_psc_ck</code></th>
<th>Counter register</th>
<th>TIF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>36</td>
</tr>
</tbody>
</table>

**42.4.7 Capture/compare channels**

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

*Figure 568* to *Figure 571* give an overview of one Capture/Compare channel.

The input stage samples the corresponding `tim_tix` input to generate a filtered signal `tim_tixf`. Then, an edge detector with polarity selection generates a signal (`tim_tixfpy`) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (`ICxPS`).
The output stage generates an intermediate waveform which is then used for reference: `tim_ocxref` (active high). The polarity acts at the end of the chain.

**Figure 569. Capture/compare channel 1 main circuit**
42.4.8 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding TIMx_ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was
already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to ‘0’ or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:1] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let’s imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on tim_ti1 when 8 consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing CC1P bit to 0 in the TIMx_CCR1 register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

**Note:** IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.
42.4.9 **PWM input mode (only for TIM15)**

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIMx_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges)

This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These 2 tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to ‘10’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.
42.4.10 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP=0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

42.4.11 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP...
bit in the TIMx_CCER register). The output pin can keep its level (OCxM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 573.
42.4.12 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are capable of upcounting only. Refer to Upcounting mode on page 1778.
In the following example, we consider PWM mode 1. The reference PWM signal \texttt{tim\_ocxref} is high as long as \texttt{TIMx\_CNT} < \texttt{TIMx\_CCRx} else it becomes low. If the compare value in \texttt{TIMx\_CCRx} is greater than the auto-reload value (in \texttt{TIMx\_ARR}) then \texttt{tim\_ocxref} is held at ‘1’. If the compare value is 0 then \texttt{tim\_ocxref} is held at ‘0’. \textit{Figure 574} shows some edge-aligned PWM waveforms in an example where \texttt{TIMx\_ARR}=8.

\textbf{Figure 574. Edge-aligned PWM waveforms (ARR=8)}

| Counter register | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 1 |
| \texttt{tim\_ocxref} |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{CCRx = 4} |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{CCxIF} | CMS=01 | CMS=10 | CMS=11 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{tim\_ocxref} |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{CCRx = 7} |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{CCxIF} | CMS=10 or 11 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{tim\_ocxref} |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{CCRx = 8} |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{CCxIF} | CMS=01 | CMS=10 | CMS=11 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{tim\_ocxref} |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{CCRx > 8} |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{CCxIF} | CMS=01 | CMS=10 | CMS=11 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{tim\_ocxref} |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{CCRx = 0} |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{CCxIF} | CMS=01 | CMS=10 | CMS=11 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| \texttt{tim\_ocxref} |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

\textbf{Dithering mode}

The PWM mode effective resolution can be increased by enabling the dithering mode, using the \texttt{DITHEN} bit in the \texttt{TIMx\_CR1} register. This applies to both the \texttt{CCR} (for duty cycle resolution increase) and \texttt{ARR} (for PWM frequency resolution increase).

The operating principle is to have the actual \texttt{CCR} (or \texttt{ARR}) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The \textit{Figure 575} below presents the dithering principle applied to 4 consecutive PWM cycles.
When the dithering mode is enabled, the register coding is changed as following (see Figure 576 for example):

- the 4 LSBs are coding for the enhanced resolution part (fractional part)
- the MSBs are left-shifted to the bits 19:4 and are coding for the base value.

**Note:** The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

The minimum frequency is given by the following formula:

\[
\text{Resolution} = \frac{F_{\text{Tim}}}{F_{\text{pwm}}} \Rightarrow F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{\max_{\text{Resolution}}}
\]
Dithering mode disabled: \[ F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65536} \]

Dithering mode enabled: \[ F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65535 + \frac{15}{16}} \]

**Note:** The maximum TIMx ARR and TIMxCCRx values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

As shown on the **Figure 577** below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

**Figure 577. PWM resolution vs frequency**

The duty cycle and/or period changes are spread over 16 consecutive periods, as described in the **Figure 578** below.
The auto-reload and compare values increments are spread following specific patterns described in the Table 440 below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 440. CCR and ARR register change dithering pattern

<table>
<thead>
<tr>
<th>LSB value</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0001</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>0010</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>0011</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>0100</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>0101</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>0110</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>0111</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>1000</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>1001</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>1010</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>1011</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>1100</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>1101</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
</tbody>
</table>
42.4.13 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

**Note:** The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

*Figure 579* represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
42.4.14 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to Table 447: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) on page 1859 for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates 2 outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

**Figure 580. Complementary output with symmetrical dead-time insertion.**

![Diagram showing complementary output with symmetrical dead-time insertion.](image)

The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on Figure 581.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must be not modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

**Note:** If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.
The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to Section 42.8.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 1863 for delay calculation.
**Re-directing tim_ocxref to tim_ocx or tim_ocxn**

In output mode (forced, output compare or PWM), tim_ocxref can be re-directed to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

*Note:* When only tim_ocxn is enabled (CCxE=0, CCxNE=1), it is *not* complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP=0 then tim_ocx=tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE=CCxNE=1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

**42.4.15 Using the break function**

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (clock failure, ECC / parity errors,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register is used to enable /disable the outputs by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shutdown level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to Table 447: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) on page 1859 for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.
The break is generated by the `tim_brk` inputs which has:
- Programmable polarity (BKP bit in the TIMx_BDTR register)
- Programmable enable bit (BKE bit in the TIMx_BDTR register)
- Programmable filter (BKF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 register.

The sources for break (`tim_brk`) channel are:
- External sources connected to one of the TIM_BKIN pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:
  - coming from a `tim_brk_cmpx` input (refer to Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals for product specific implementation)
  - coming from a system break request on the `tim_sys_brk` inputs (refer to Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals for product specific implementation)

Break events can also be generated by software using BG bit in the TIMx_EGR register. All sources are ORed before entering the timer `tim_brk` inputs, as per Figure 584 below.

![Figure 584. Break circuitry overview](image)

**Caution:** An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.
When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.

- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the GPIO) else the enable output remains high.

- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 tim_ker_ck clock cycles).
  - If OSSI=0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.

- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the tim_brk input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR Register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to Section 42.8.14: TIMx break and dead-time register (TIMx_BDTR) (x = 16 to 17). The LOCK bits can be written only once after an MCU reset.

The Figure 585 shows an example of behavior of the outputs in response to a break.
Figure 585. Output behavior in response to a break event on tim_brk

- **tim_ocxref**
- **tim_ocx** (tim_ocxn not implemented, CCxP=0, OISx=1)
- **tim_ocx** (tim_ocxn not implemented, CCxP=0, OISx=0)
- **tim_ocx** (tim_ocxn not implemented, CCxP=1, OISx=1)
- **tim_ocx** (tim_ocxn not implemented, CCxP=1, OISx=0)
- **tim_ocx**
- **tim_ocxn** (CCxE=1, CCxP=0, OISx=0, CCxNE=1, CCxNP=0, OISxN=1)
- **tim_ocxn** (CCxE=1, CCxP=0, OISx=1, CCxNE=1, CCxNP=1, OISxN=1)
- **tim_ocxn** (CCxE=1, CCxP=0, OISx=0, CCxNE=0, CCxNP=0, OISxN=1)
- **tim_ocxn** (CCxE=1, CCxP=0, OISx=1, CCxNE=0, CCxNP=0, OISxN=0)
- **tim_ocxn** (CCxE=1, CCxP=0, CCxNE=0, CCxNP=0, OISx=OISxN=0 or OISx=OISxN=1)

BREAK (MOE)
42.4.16 Bidirectional break input

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on Figure 586.

They are used to have:
- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The tim_brk input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKinP and BKP bits). Any break request coming either from system (for example CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (triggered by setting the BG bit) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKin I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:
- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see Table 441)

<table>
<thead>
<tr>
<th>MOE</th>
<th>BKBID</th>
<th>BKDSRM</th>
<th>Break protection state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>Armed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Armed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Disarmed</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Armed</td>
</tr>
</tbody>
</table>

Arming and re-arming break circuitry

The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).
The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

**Figure 586. Output redirection**

---

### 42.4.17 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int input can be selected among several inputs, as shown on Figure 587 below.
42.4.18 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The Figure 588 describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.
Figure 588. 6-step generation, COM example (OSSR=1)

Example 1

- **Counter (CNT)**
  - (CCRx)
- **tim_ocxref**
- **COM event**
  - CCxE = 1
  - CCxNE = 0
  - OCxM = 0010 (forced inactive)
  - Write COM to 1
- **Example 1**
  - **tim_ocx**
  - **tim_ocxn**

Example 2

- **Counter (CNT)**
  - (CCRx)
- **tim_ocxref**
- **COM event**
  - CCxE = 1
  - CCxNE = 0
  - OCxM = 0100 (forced inactive)
  - Write CCxNE to 1 and OCxM to 0101
- **Example 2**
  - **tim_ocx**
  - **tim_ocxn**

Example 3

- **Counter (CNT)**
  - (CCRx)
- **tim_ocxref**
- **COM event**
  - CCxE = 1
  - CCxNE = 0
  - OCxM = 0010 (forced inactive)
  - Write CCxNE to 0 and OCxM to 0100
- **Example 3**
  - **tim_ocx**
  - **tim_ocxn**
42.4.19 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- $\text{CNT} < \text{CCR}_x \leq \text{ARR}$ (in particular, $0 < \text{CCR}_x$)

![Figure 589. Example of one pulse mode.](MSv62344V1)

For example one may want to generate a positive pulse on TIM_OC1 with a length of $t_{\text{PULSE}}$ and after a delay of $t_{\text{DELAY}}$ as soon as a positive edge is detected on the TIM_TI2 input pin.

Let's use TIM_TI2FP2 as trigger 1:

1. Select the proper TIM_TI2_IN[15:1] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TIM_TI2FP2 to TIM_TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. TIM_TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TIM_TI2FP2 as trigger for the slave mode controller (TIM_TRGI) by writing TS='00110' in the TIMx_SMCR register.
5. TIM_TI2FP2 is used to start the counter by writing SMS to ‘110’ in the TIMx_SMCR register (trigger mode).
The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The $t_{\text{DELAY}}$ is defined by the value written in the TIMx_CCR1 register.
- The $t_{\text{PULSE}}$ is defined by the difference between the auto-reload value and the compare value ($\text{TIMx.ARR} - \text{TIMx.CCR1}$).
- Let’s say one want to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=’1’ in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to ‘0’ in this example.

Since only 1 pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay $t_{\text{DELAY.min}}$ we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 42.4.20 Retriggerable one pulse mode (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in Section 42.4.19:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for Retrigerrable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

**Note:** The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

*This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.*
42.4.21 UIF bit remapping

The UIFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

42.4.22 Timer input XOR function (TIM15 only)

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in Figure 591.

42.4.23 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.
The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

**Slave mode: Reset mode**

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P='0' and CC1NP='0' in the TIMx_CCRER register to validate the polarity (and detect rising edges only).

2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.

3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

**Figure 592. Control circuit in reset mode**
**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when `tim_ti1` input is low:

1. Configure the channel 1 to detect low levels on `tim_ti1`. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP = '0' in the TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select `tim_ti1` as the input source by writing TS=00101 in TIMx_SMCR register.

3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as `tim_ti1` is low and stops as soon as `tim_ti1` becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on `tim_ti1` and the actual stop of the counter is due to the resynchronization circuit on `tim_ti1` input.

**Figure 593. Control circuit in gated mode**

![Control circuit in gated mode](image-url)
Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P='1' and CC2NP='0' in the TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS=110 in the TIMx_SMCR register. Select tim_ti2 as the input source by writing TS=00110 in the TIMx_SMCR register. When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 594. Control circuit in trigger mode

Slave mode selection preload for run-time update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

42.4.24 Slave mode – combined reset + trigger mode (TIM15 only)

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

42.4.25 Slave mode – combined reset + gated mode (TIM15 only)

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).
42.4.26 Timer synchronization (TIM15 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to Section 39.4.23: Timer synchronization for details.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

42.4.27 Using timer output as trigger for other timers (TIM16/TIM17 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination’s timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

42.4.28 ADC triggers (TIM15 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

42.4.29 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to re-program several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see Section 42.8.19: TIMx DMA control register (TIMx_DCR)(x = 16 to 17) for details).

For example, the timer DMA burst feature can be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:
   - DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

**Note:** A null value can be written to the reserved registers.

### 42.4.30 TIM15/TIM16/TIM17 DMA requests

The TIM15/TIM16/TIM17 can generate a DMA requests, as shown in Table 442.

<table>
<thead>
<tr>
<th>DMA request signal</th>
<th>DMA request</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_upd_dma</td>
<td>Update</td>
<td>UDE</td>
</tr>
<tr>
<td>tim_cc1_dma</td>
<td>Capture/compare 1</td>
<td>CC1DE</td>
</tr>
<tr>
<td>tim_com_dma(1)</td>
<td>Commutation (COM)</td>
<td>COMDE</td>
</tr>
<tr>
<td>tim_trg_dma(1)</td>
<td>Trigger</td>
<td>TDE</td>
</tr>
</tbody>
</table>

1. Available for TIM15 only.

### 42.4.31 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop.
The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

For more details, refer to the debug section.

### 42.5 TIM15/TIM16/TIM17 low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The timer operation is stopped and the register content is kept. No interrupt can be generated.</td>
</tr>
<tr>
<td>Standby</td>
<td>The timer is powered-down and must be reinitialized after exiting the Standby mode.</td>
</tr>
</tbody>
</table>

### 42.6 TIM15/TIM16/TIM17 interrupts

The TIM15/TIM16/TIM17 can generate multiple interrupts, as shown in Table 444.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop and Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM</td>
<td>Update</td>
<td>UIF</td>
<td>UIE</td>
<td>write 0 in UIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 1</td>
<td>CC1IF</td>
<td>CC1IE</td>
<td>write 0 in CC1IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 2(1)</td>
<td>CC2IF</td>
<td>CC2IE</td>
<td>write 0 in CC2IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Commutation (COM)</td>
<td>COMIF</td>
<td>COMIE</td>
<td>write 0 in COMIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Trigger(1)</td>
<td>TIF</td>
<td>TIE</td>
<td>write 0 in TIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Break</td>
<td>BIF</td>
<td>BIE</td>
<td>write 0 in BIF</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

1. Available for TIM15 only.
42.7 **TIM15 registers**

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

### 42.7.1 TIM15 control register 1 (TIM15_CR1)

Address offset: 0x00  
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:13 Reserved, must be kept at reset value.

- **Bit 12 DITHEN**: Dithering enable  
  0: Dithering disabled  
  1: Dithering enabled  
  *Note: The DITHEN bit can only be modified when CEN bit is reset.*

- **Bit 11 UIFREMAP**: UIF status bit remapping  
  0: No remapping. UIF status bit is not copied to TIM15_CNT register bit 31.  
  1: Remapping enabled. UIF status bit is copied to TIM15_CNT register bit 31.

- **Bit 10 Reserved, must be kept at reset value.**

- **Bits 9:8 CKD[1:0]**: Clock division  
  This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (tDTS) used by the dead-time generators and the digital filters (tim_tix)  
  00: tDTS = tim_ker_ck  
  01: tDTS = 2*tim_ker_ck  
  10: tDTS = 4*tim_ker_ck  
  11: Reserved

- **Bit 7 ARPE**: Auto-reload preload enable  
  0: TIM15_ARR register is not buffered  
  1: TIM15_ARR register is buffered

- **Bits 6:4 Reserved, must be kept at reset value.**

- **Bit 3 OPM**: One-pulse mode  
  0: Counter is not stopped at update event  
  1: Counter stops counting at the next update event (clearing the bit CEN)
Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt if enabled. These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

*Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

### 42.7.2 TIM15 control register 2 (TIM15_CR2)

Address offset: 0x04

Reset value: 0x0000

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **OIS2**: Output idle state 2 (tim_oc2 output)

0: \( \text{tim\_oc2}=0 \) when MOE=0
1: \( \text{tim\_oc2}=1 \) when MOE=0

*Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM15_BKR register).*

Bit 9 **OIS1N**: Output idle state 1 (tim_oc1n output)

0: \( \text{tim\_oc1n}=0 \) after a dead-time when MOE=0
1: \( \text{tim\_oc1n}=1 \) after a dead-time when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).*

Bit 8 **OIS1**: Output idle state 1 (tim_oc1 output)

0: \( \text{tim\_oc1}=0 \) after a dead-time when MOE=0
1: \( \text{tim\_oc1}=1 \) after a dead-time when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).*
Bit 7  **TI1S**: tim_ti1 selection
0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 6:4  **MMS[2:0]**: Master mode selection
These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:
000:  **Reset** - the UG bit from the TIM15_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.
001:  **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM15_SMCR register).
010:  **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.
011:  **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).
100:  **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo).
101:  **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo).

Bit 3  **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2  **CCUS**: Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi.

*Note: This bit acts only on channels that have a complementary output.*

Bit 1  Reserved, must be kept at reset value.

Bit 0  **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

*Note: This bit acts only on channels that have a complementary output.*
### 42.7.3 TIM15 slave mode control register (TIM15_SMCR)

**Address offset:** 0x08  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:25</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>SMSPE:</td>
<td>SMS preload enable</td>
</tr>
<tr>
<td></td>
<td>Bit 24</td>
<td>This bit selects whether the SMS[3:0] bitfield is preloaded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: SMS[3:0] bitfield is not preloaded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: SMS[3:0] preload is enabled.</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19:17</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15:8</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>7</td>
<td>MSM:</td>
<td>Master/slave mode</td>
</tr>
<tr>
<td></td>
<td>Bit 7</td>
<td>0: No action</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: The effect of an event on the trigger input (tim_trgi) is delayed to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>allow a perfect synchronization between the current timer and its slaves</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(through tim_trgo). It is useful if we want to synchronize several timers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>on a single external event.</td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded.

0: SMS[3:0] bitfield is not preloaded  
1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM**: Master/slave mode

0: No action  
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.
Bits 21, 20, 6, 5, 4  **TS[4:0]: Trigger selection**

This bit field selects the trigger input to be used to synchronize the counter.

- 00000: Internal Trigger 0 (tim_itr0)
- 00001: Internal Trigger 1 (tim_itr1)
- 00010: Internal Trigger 2 (tim_itr2)
- 00011: Internal Trigger 3 (tim_itr3)
- 00100: tim_ti1 Edge Detector (tim_ti1f_ed)
- 00101: Filtered Timer Input 1 (tim_ti1fp1)
- 00110: Filtered Timer Input 2 (tim_ti2fp2)
- 00111: Reserved
- 01000: Internal Trigger 4 (tim_itr4)
- 01001: Internal Trigger 5 (tim_itr5)
- 01010: Internal Trigger 6 (tim_itr6)
- 01011: Internal Trigger 7 (tim_itr7)
- 01100: Internal Trigger 8 (tim_itr8)
- 01101: Internal Trigger 9 (tim_itr9)
- 01110: Internal Trigger 10 (tim_itr10)
- 10000: Internal trigger 12 (tim_itr12)
- 10001: Internal trigger 13 (tim_itr13)
- 10010: Internal trigger 14 (tim_itr14)
- 10011: Internal trigger 15 (tim_itr15)

Others: Reserved

See [Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals](#) for more details on **tim_itrx** meaning for each timer.

*Note:* These bits must be changed only when they are not used (for example when **SMS=000**) to avoid wrong edge detections at the transition.

Bit 3  **Reserved**, must be kept at reset value.
Bits 16, 2, 1, 0  **SMS[3:0]: Slave mode selection**

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.
0001: Reserved
0010: Reserved
0011: Reserved
0100: Reset Mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.
0101: Gated Mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110: Trigger Mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.
0111: External Clock Mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.
1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
Others: Reserved.

**Note:** The gated mode must not be used if tim_t1f_ed is selected as the trigger input (TS='00100'). Indeed, tim_t1f_ed outputs 1 pulse for each transition on tim_t1f, whereas the gated mode checks the level of the trigger signal.

**Note:** The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

### 42.7.4 TIM15 DMA/interrupt enable register (TIM15_DIER)

Address offset: 0x0C

Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDE</td>
<td>COMDE</td>
<td>UDE</td>
<td>BIE</td>
<td>TIE</td>
<td>COMIE</td>
<td>UIE</td>
<td>CC1DE</td>
<td>UDE</td>
<td>BIE</td>
<td>TIE</td>
<td>COMIE</td>
<td>UIE</td>
<td>CC1DE</td>
<td>UDE</td>
<td>BIE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 15  Reserved, must be kept at reset value.

Bit 14  **TDE:** Trigger DMA request enable

0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13  **COMDE:** COM DMA request enable

0: COM DMA request disabled
1: COM DMA request enabled

Bits 12:10  Reserved, must be kept at reset value.
**42.7.5 TIM15 status register (TIM15_SR)**

Address offset: 0x10

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit (15:0)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>10</td>
<td>CC2OF: Capture/Compare 2 overcapture flag</td>
</tr>
<tr>
<td></td>
<td>Refer to CC1OF description</td>
</tr>
<tr>
<td>9</td>
<td>CC1OF: Capture/Compare 1 overcapture flag</td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware only when the</td>
</tr>
<tr>
<td></td>
<td>corresponding channel is configured in input</td>
</tr>
<tr>
<td></td>
<td>capture mode. It is cleared by software by</td>
</tr>
<tr>
<td></td>
<td>writing it to '0'.</td>
</tr>
<tr>
<td></td>
<td>0: No overcapture has been detected</td>
</tr>
<tr>
<td></td>
<td>1: The counter value has been captured in</td>
</tr>
<tr>
<td></td>
<td>TIM15_CCR1 register while CC1IF flag was</td>
</tr>
<tr>
<td></td>
<td>already set</td>
</tr>
<tr>
<td>8</td>
<td>UIF: Update interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Update interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Update interrupt enabled</td>
</tr>
<tr>
<td>7</td>
<td>BIF: Trigger interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Trigger interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Trigger interrupt enabled</td>
</tr>
<tr>
<td>6</td>
<td>COMIF: COM interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: COM interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: COM interrupt enabled</td>
</tr>
<tr>
<td>5</td>
<td>CC1IE: Capture/Compare 1 interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: CC1 interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: CC1 interrupt enabled</td>
</tr>
<tr>
<td>4</td>
<td>UIE: Update interrupt enable</td>
</tr>
<tr>
<td>3</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>2</td>
<td>CC2IE: Capture/Compare 2 interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: CC2 interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: CC2 interrupt enabled</td>
</tr>
<tr>
<td>1</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>0</td>
<td>Capture/Compare 1 DMA request enable</td>
</tr>
<tr>
<td></td>
<td>0: CC1 DMA request disabled</td>
</tr>
<tr>
<td></td>
<td>1: CC1 DMA request enabled</td>
</tr>
<tr>
<td></td>
<td>Update DMA request enable</td>
</tr>
<tr>
<td></td>
<td>0: Update DMA request disabled</td>
</tr>
<tr>
<td></td>
<td>1: Update DMA request enabled</td>
</tr>
</tbody>
</table>

Bits 15:11: Reserved, must be kept at reset value.
Bit 7 BIF: Break interrupt flag
   This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
   0: No break event occurred
   1: An active level has been detected on the break input

Bit 6 TIF: Trigger interrupt flag
   This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
   0: No trigger event occurred
   1: Trigger interrupt pending

Bit 5 COMIF: COM interrupt flag
   This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
   0: No COM event occurred
   1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 CC2IF: Capture/Compare 2 interrupt flag
   refer to CC1IF description

Bit 1 CC1IF: Capture/Compare 1 interrupt flag
   This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
   0: No compare match / No input capture occurred
   1: A compare match or an input capture occurred

   If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

   If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 UIF: Update interrupt flag
   This bit is set by hardware on an update event. It is cleared by software.
   0: No update occurred.
   1: Update interrupt pending. This bit is set by hardware when the registers are updated:
      – At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIM15_CR1 register.
      – When CNT is reinitialized by software using the UG bit in TIM15_EGR register, if URS=0 and UDIS=0 in the TIM15_CR1 register.
      – When CNT is reinitialized by a trigger event (refer to Section 42.7.3: TIM15 slave mode control register (TIM15_SMCR)), if URS=0 and UDIS=0 in the TIM15_CR1 register.
42.7.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:8</th>
<th>Bits 5:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>BG: Break generation</td>
</tr>
<tr>
<td>Bit 15:8 Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>Bit 7</td>
<td>BG: Break generation</td>
</tr>
<tr>
<td>This bit is set by software in order to generate an event, it is automatically cleared by hardware.</td>
<td></td>
</tr>
<tr>
<td>0: No action</td>
<td></td>
</tr>
<tr>
<td>1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.</td>
<td></td>
</tr>
<tr>
<td>Bit 6</td>
<td>TG: Trigger generation</td>
</tr>
<tr>
<td>This bit is set by software in order to generate an event, it is automatically cleared by hardware.</td>
<td></td>
</tr>
<tr>
<td>0: No action</td>
<td></td>
</tr>
<tr>
<td>1: The TIF flag is set in TIM15_SR register. Related interrupt or DMA transfer can occur if enabled</td>
<td></td>
</tr>
<tr>
<td>Bit 5</td>
<td>COMG: Capture/Compare control update generation</td>
</tr>
<tr>
<td>This bit can be set by software, it is automatically cleared by hardware.</td>
<td></td>
</tr>
<tr>
<td>0: No action</td>
<td></td>
</tr>
<tr>
<td>1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits</td>
<td></td>
</tr>
<tr>
<td>Note: This bit acts only on channels that have a complementary output.</td>
<td></td>
</tr>
<tr>
<td>Bit 4:3 Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>Bit 2</td>
<td>CC2G: Capture/Compare 2 generation</td>
</tr>
<tr>
<td>Refer to CC1G description</td>
<td></td>
</tr>
<tr>
<td>Bit 1</td>
<td>CC1G: Capture/Compare 1 generation</td>
</tr>
<tr>
<td>This bit is set by software in order to generate an event, it is automatically cleared by hardware.</td>
<td></td>
</tr>
<tr>
<td>0: No action</td>
<td></td>
</tr>
<tr>
<td>1: A capture/compare event is generated on channel 1:</td>
<td></td>
</tr>
<tr>
<td>If channel CC1 is configured as output:</td>
<td></td>
</tr>
<tr>
<td>CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.</td>
<td></td>
</tr>
<tr>
<td>If channel CC1 is configured as input:</td>
<td></td>
</tr>
<tr>
<td>The current value of the counter is captured in TIM15_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.</td>
<td></td>
</tr>
<tr>
<td>Bit 0</td>
<td>UG: Update generation</td>
</tr>
<tr>
<td>This bit can be set by software, it is automatically cleared by hardware.</td>
<td></td>
</tr>
<tr>
<td>0: No action</td>
<td></td>
</tr>
<tr>
<td>1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).</td>
<td></td>
</tr>
</tbody>
</table>
42.7.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

### Input capture mode

| Bit 31:16 | Reserved, must be kept at reset value. |
| Bit 15:12 | IC2F[3:0]: Input capture 2 filter |
| Bit 11:10 | IC2PSC[1:0]: Input capture 2 prescaler |
| Bit 9:8   | CC2S[1:0]: Capture/Compare 2 selection |

This bit-field defines the direction of the channel (input/output) as well as the used input.

- 00: CC2 channel is configured as output
- 01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
- 10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
- 11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

**Note:** CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIM15_CCER).
42.7.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).
Output compare mode:

Bits 31:25  Reserved, must be kept at reset value.

Bits 23:17  Reserved, must be kept at reset value.

Bit 15  **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12  **OC2M[3:0]**: Output compare 2 mode

Bit 11  **OC2PE**: Output compare 2 preload enable

Bit 10  **OC2FE**: Output compare 2 fast enable

Bits 9:8  **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM15_SMCR register).

**Note**: **CC2S** bits are writable only when the channel is OFF (**CC2E = '0'** in **TIM15_CCER**).

Bit 7  **OC1CE**: Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input.
Bits 16, 6:4 **OC1M[3:0]:** Output compare 1 mode

These bits define the behavior of the output reference signal `tim_oc1ref` from which `tim_oc1` and `tim_oc1n` are derived. `tim_oc1ref` is active high whereas `tim_oc1` and `tim_oc1n` active level depends on `CC1P` and `CC1NP` bits.

0000: Frozen - The comparison between the output compare register TIM15_CCR1 and the counter TIM15_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. `tim_oc1ref` signal is forced high when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0010: Set channel 1 to inactive level on match. `tim_oc1ref` signal is forced low when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0011: Toggle - `tim_oc1ref` toggles when TIM15_CNT=TIM15_CCR1.

0100: Force inactive level - `tim_oc1ref` is forced low.

0101: Force active level - `tim_oc1ref` is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIM15_CNT<TIM15_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIM15_CNT<TIM15_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on `tim_trgi` signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on `tim_trgi` signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on `tim_trgi` signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on `tim_trgi` signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved

1011: Reserved

1100: Combined PWM mode 1 - `tim_oc1ref` has the same behavior as in PWM mode 1. `tim_oc1ref` is the logical OR between `tim_oc1ref` and `tim_oc2ref`.

1101: Combined PWM mode 2 - `tim_oc1ref` has the same behavior as in PWM mode 2. `tim_oc1ref` is the logical AND between `tim_oc1ref` and `tim_oc2ref`.

1110: Reserved

1111: Reserved,

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S='00' (the channel is configured in output). In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

On channels that have a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.
Bit 3 **OC1PE**: Output Compare 1 preload enable

- 0: Preload register on TIM15_CCR1 disabled. TIM15_CCR1 can be written at anytime, the new value is taken in account immediately.
- 1: Preload register on TIM15_CCR1 enabled. Read/Write operations access the preload register. TIM15_CCR1 preload value is loaded in the active register at each update event.

*Note*: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S='00' (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

- 0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
- 1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

- 00: CC1 channel is configured as output.
- 01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.
- 10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.
- 11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

*Note*: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIM15_CER).

### 42.7.9 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:8 Reserved, must be kept at reset value.

- Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
  - Refer to CC1NP description

- Bit 6 Reserved, must be kept at reset value.

- Bit 5 **CC2P**: Capture/Compare 2 output polarity
  - Refer to CC1P description

- Bit 4 **CC2E**: Capture/Compare 2 output enable
  - Refer to CC1E description
Bit 3 CC1NP: Capture/Compare 1 complementary output polarity
CC1 channel configured as output:
0: tim_oc1n active high
1: tim_oc1n active low
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S="00" (the channel is configured in output).
On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 CC1NE: Capture/Compare 1 complementary output enable
0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 CC1P: Capture/Compare 1 output polarity
CC1 channel configured as output:
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode).
CC1NP=1, CC1P=0: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).
CC1NP=1, CC1P=1: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).
On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 CC1E: Capture/Compare 1 output enable
0: Capture mode disabled / OC1 is not active (see below)
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to Table 445 for details.
Table 445. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15)

<table>
<thead>
<tr>
<th>Control bits</th>
<th>Output states(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOE bit</td>
<td>OSSI bit</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and GPIO control and alternate function selection registers.
42.7.10 TIM15 counter (TIM15_CNT)

Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>UIFCPY: UIF Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is a read-only copy of the UIF bit in the TIM15_ISR register.</td>
</tr>
<tr>
<td>Bits 30:16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 15:0</td>
<td>CNT[15:0]: Counter value</td>
</tr>
<tr>
<td></td>
<td>Non-dithering mode (DITHEN = 0)</td>
</tr>
<tr>
<td></td>
<td>The register holds the counter value.</td>
</tr>
<tr>
<td></td>
<td>Dithering mode (DITHEN = 1)</td>
</tr>
<tr>
<td></td>
<td>The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.</td>
</tr>
</tbody>
</table>

42.7.11 TIM15 prescaler (TIM15_PSC)

Address offset: 0x28
Reset value: 0x0000

| Bit 15:0 | PSC[15:0]: Prescaler value |
|          | The counter clock frequency (f_{tim_cnt_ck}) is equal to f_{tim_psc_ck} / (PSC[15:0] + 1). |
|          | PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM15_EGR register or through trigger controller when configured in "reset mode"). |
42.7.12 TIM15 auto-reload register (TIM15_ARR)

Address offset: 0x2C
Reset value: 0x0000 FFFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]: Auto-reload value**

ARR is the value to be loaded in the actual auto-reload register.
Refer to the [Section 42.4.3: Time-base unit on page 1776](#) for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
**Non-dithering mode (DITHEN = 0)**
The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.
**Dithering mode (DITHEN = 1)**
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

42.7.13 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]: Repetition counter reload value**

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.
When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM15_RCR register is not taken in account until the next repetition update event.
It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:
- the number of PWM periods in edge-aligned mode
- the number of half PWM period in center-aligned mode
42.7.14 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34
Reset value: 0x0000 0000

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16 |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

**If channel CC1 is configured as output:**
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIM15_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

**If channel CC1 is configured as input:**
CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.
42.7.15  TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

CCR2[15:0]

Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  CCR2[19:0]: Capture/compare 2 value

If channel CC2 is configured as output:
CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIM15_CCMR2 register
(bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when
an update event occurs.
The active capture/compare register contains the value to be compared to the counter
TIM15_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)
The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the
dithered part.

If channel CC2 is configured as input:
CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The
TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)
The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

42.7.16  TIM15 break and dead-time register (TIM15_BDTR)

Address offset: 0x44
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>BKBID</td>
<td>rw</td>
<td>BK DRSM</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

MOE  AOE  BKP  BKE  OSSR  OSSI  LOCK[1:0]  DTG[7:0]
| rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  |

Note:  As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSR, OSSI and DTG[7:0] bits may
be write-locked depending on the LOCK configuration, it may be necessary to configure all
of them during the first write access to the TIM15_BDTR register.

RM0481 Rev 1 1837/3119
Bits 31:29  Reserved, must be kept at reset value.

Bit 28  **BKBID**: Break bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode

In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

*Note*: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

*Note*: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27  Reserved, must be kept at reset value.

Bit 26  **BKDSRM**: Break disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed

This bit is cleared by hardware when no break source is active.

The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled until it is reset by hardware, indicating that the fault condition has disappeared.

*Note*: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20  Reserved, must be kept at reset value.

Bits 19:16  **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample the tim_brk input signal and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: \(f_{\text{SAMPLING}} = f_{\text{tim_ker_ck}}, N = 2\)
0010: \(f_{\text{SAMPLING}} = f_{\text{tim_ker_ck}}, N = 4\)
0011: \(f_{\text{SAMPLING}} = f_{\text{tim_ker_ck}}, N = 8\)
0100: \(f_{\text{SAMPLING}} = f_{\text{DTS}/2}, N = 6\)
0101: \(f_{\text{SAMPLING}} = f_{\text{DTS}/2}, N = 8\)
0110: \(f_{\text{SAMPLING}} = f_{\text{DTS}/4}, N = 6\)
0111: \(f_{\text{SAMPLING}} = f_{\text{DTS}/4}, N = 8\)
1000: \(f_{\text{SAMPLING}} = f_{\text{DTS}/8}, N = 6\)
1001: \(f_{\text{SAMPLING}} = f_{\text{DTS}/8}, N = 8\)
1010: \(f_{\text{SAMPLING}} = f_{\text{DTS}/16}, N = 5\)
1011: \(f_{\text{SAMPLING}} = f_{\text{DTS}/16}, N = 6\)
1100: \(f_{\text{SAMPLING}} = f_{\text{DTS}/16}, N = 8\)
1101: \(f_{\text{SAMPLING}} = f_{\text{DTS}/32}, N = 5\)
1110: \(f_{\text{SAMPLING}} = f_{\text{DTS}/32}, N = 6\)
1111: \(f_{\text{SAMPLING}} = f_{\text{DTS}/32}, N = 8\)

*Note*: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).
Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_ocx and tim_ocxn outputs are disabled or forced to idle state depending on the OSSI bit.
1: tim_ocx and tim_ocxn outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM15_CCER register)

See tim_ocx/tim_ocxn enable description for more details (Section 42.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1831).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software
1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low
1: Break input tim_brk is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk clock failure event) disabled
1: Break inputs (tim_brk and tim_sys_brk clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_ocx/tim_ocxn enable description for more details (Section 42.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1831).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)
1: When inactive, tim_ocx/tim_ocxn outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).*

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See tim_ocx/tim_ocxn enable description for more details (Section 42.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1831).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (tim_ocx/tim_ocxn enable output signal=0)
1: When inactive, tim_ocx/tim_ocxn outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. tim_ocx/tim_ocxn enable output signal=1)

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).*
42.7.17 TIM15 timer deadtime register 2 (TIM15_DTR2)

Address offset: 0x054
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>DTPE</td>
<td>DTAE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable
0: Deadtime value is not preloaded
1: Deadtime value preload is enabled

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).*
Bit 16 **DTAE**: Deadtime asymmetric enable
- 0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register
- 1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

*Note*: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

- \( DTGF[7:5]=0x0 \Rightarrow DTF=DTGF[7:0] \times t_{dtg} = t_{DTS} \)
- \( DTGF[7:5]=10x \Rightarrow DTF=(64+DTGF[5:0]) \times t_{dtg} = 2t_{DTS} \)
- \( DTGF[7:5]=110 \Rightarrow DTF=(32+DTGF[4:0]) \times t_{dtg} = 8t_{DTS} \)
- \( DTGF[7:5]=111 \Rightarrow DTF=(32+DTGF[4:0]) \times t_{dtg} = 16t_{DTS} \)

Example if \( T_{DTS}=125\text{ns} \)(8MHz), dead-time possible values are:
- 0 to 15875 ns by 125 ns steps,
- 16 us to 31750 ns by 250 ns steps,
- 32 us to 63us by 1 us steps,
- 64 us to 126 us by 2 us steps

*Note*: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

### 42.7.18 TIM15 input selection register (TIM15_TISEL)

Address offset: 0x5C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input
- 0000: TIM15_CH2 input (tim_ti2_in0)
- 0001: tim_ti2_in1
- ...
- 1111: tim_ti2_in15

Refer to Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
- 0000: TIM15_CH1 input (tim_ti1_in0)
- 0001: tim_ti1_in1
- ...
- 1111: tim_ti1_in15

Refer to Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals for interconnects list.
42.7.19 TIM15 alternate function register 1 (TIM15_AF1)

Address offset: 0x060
Reset value: 0x0000 0001

Refer to Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 BKCMP4P: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input is active high
1: tim_brk_cmp4 input is active low
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 12 BKCMP3P: tim_brk_cmp3 input polarity
This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp3 input is active high
1: tim_brk_cmp3 input is active low
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 11 BKCMP2P: tim_brk_cmp2 input polarity
This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp2 input is active high
1: tim_brk_cmp2 input is active low
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 BKCMP1P: tim_brk_cmp1 input polarity
This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp1 input is active high
1: tim_brk_cmp1 input is active low
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).
Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high
1: TIMx_BKIN input is active low

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer’s tim_brk input. mdf_brkx output is ‘ORed’ with the other tim_brk sources.

0: tim_brk_cmp8 input disabled
1: tim_brk_cmp8 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer’s tim_brk input. COMP7 output is ‘ORed’ with the other tim_brk sources.

0: tim_brk_cmp7 input disabled
1: tim_brk_cmp7 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer’s tim_brk input. tim_brk_cmp6 output is ‘ORed’ with the other tim_brk sources.

0: tim_brk_cmp6 input disabled
1: tim_brk_cmp6 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer’s tim_brk input. tim_brk_cmp5 output is ‘ORed’ with the other tim_brk sources.

0: tim_brk_cmp5 input disabled
1: tim_brk_cmp5 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer’s tim_brk input. tim_brk_cmp4 output is ‘ORed’ with the other tim_brk sources.

0: tim_brk_cmp4 input disabled
1: tim_brk_cmp4 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer’s tim_brk input. tim_brk_cmp3 output is ‘ORed’ with the other tim_brk sources.

0: tim_brk_cmp3 input disabled
1: tim_brk_cmp3 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*
42.7.20 TIM15 alternate function register 2 (TIM15_AF2)

Address offset: 0x064
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18-16</td>
<td>OCRSEL[2:0]: ocref_clr source selection</td>
</tr>
<tr>
<td></td>
<td>These bits select the ocref_clr input source.</td>
</tr>
<tr>
<td></td>
<td>000: tim_ocref_clr0</td>
</tr>
<tr>
<td></td>
<td>001: tim_ocref_clr1</td>
</tr>
<tr>
<td></td>
<td>010: tim_ocref_clr2</td>
</tr>
<tr>
<td></td>
<td>011: tim_ocref_clr3</td>
</tr>
<tr>
<td></td>
<td>100: tim_ocref_clr4</td>
</tr>
<tr>
<td></td>
<td>101: tim_ocref_clr5</td>
</tr>
<tr>
<td></td>
<td>110: tim_ocref_clr6</td>
</tr>
<tr>
<td></td>
<td>111: tim_ocref_clr7</td>
</tr>
<tr>
<td></td>
<td>Refer to Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals for product specific implementation.</td>
</tr>
<tr>
<td></td>
<td>Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).</td>
</tr>
</tbody>
</table>

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).
### 42.7.21 TIM15 DMA control register (TIM15_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DBS[3:0]: DMA burst source selection**

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

- 0000: Reserved
- 0001: Update
- 0010: CC1
- 0110: COM
- 0111: Trigger
- Other: reserved

**DBL[4:0]: DMA burst length**

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM15_DMAR address).

- 00000: 1 transfer,
- 00001: 2 transfers,
- 00010: 3 transfers,
- ...
- 10001: 18 transfers.

**DBA[4:0]: DMA base address**

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIM15_DMAR address). DBA is defined as an offset starting from the address of the TIM15_CR1 register.

Example:

- 00000: TIM15_CR1,
- 00001: TIM15_CR2,
- 00010: TIM15_SMCR,
- ...
### 42.7.22 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x3E0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Offset</th>
<th>Bits 31:0 DMAR[31:0]: DMA register for burst accesses</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A read or write operation to the DMAR register accesses the register located at the address TIM15_CR1 address + (DBA + DMA index) x 4 where TIM15_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM15_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM15_DCR).</td>
</tr>
</tbody>
</table>

### 42.7.23 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits 31:20</th>
<th>Bits 19:16</th>
<th>Bits 15:12</th>
<th>Bits 11:8</th>
<th>Bits 7:4</th>
<th>Bits 3:0</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>TIM15_CR1</td>
<td>UIFREMA</td>
<td>CKD[1:0]</td>
<td>ARPE</td>
<td>OPM</td>
<td>URS</td>
<td>UDS</td>
<td>0x0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>TIM15_CR2</td>
<td>OIS2</td>
<td>OIS1N</td>
<td>OIS1</td>
<td>TI1S</td>
<td>MMS[2:0]</td>
<td>CCDS</td>
<td>0x0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>TIM15_DIER</td>
<td>TIE</td>
<td>CO1IE</td>
<td>CO2IE</td>
<td>CC1IE</td>
<td>CO2IE</td>
<td>CC1IE</td>
<td>0x0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>TIM15_SR</td>
<td>BIF</td>
<td>TIF</td>
<td>COMIF</td>
<td>CC2IF</td>
<td>CC1IF</td>
<td>UG</td>
<td>0x0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>TIM15_EGR</td>
<td>BG</td>
<td>TG</td>
<td>CO2G</td>
<td>CC2G</td>
<td>CC1G</td>
<td>UG</td>
<td>0x0000</td>
</tr>
</tbody>
</table>
0x54

0x58

TIM15_DTR2

0

0

0

0

Reserved

RM0481 Rev 1

Reserved

0

0

OSSI

0

0

0

0

0

0

0

0

Res.

Res.

Res.

LOCK
[1:0]

Res.

0

OSSR

BKF[3:0]

Res.

Reserved

BKE

0

Res.

0

BKP

0
0

AOE

0
0

Res.

0
0
0
1
1
1
1
1
1
1
1

Res.
Res.
Res.
Res.
Res.
Res.
Res.
Res.
Res.
Res.
Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Reset value

Res.

0

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

UIFCPY or Res.

Res.

Res.

OC1M[3]
OC2CE

OC2PE
OC2FE

0
0
0
0
0
0
0
0

Res.
Res.
Res.
Res.
Res.
Res.
Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

OC2M[3]

Res.

Res.

Res.

Res.

Res.

Res.

Res.

CC2S
[1:0]

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

0

MOE

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Reset value

Res.

Reset value
0

DTAE

Res.

Res.

Res.

Res.

Res.

Res.

0

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Reset value

Res.

Res.

Res.

Res.

Res.

0

BKDSRM

Res.

Res.

Res.

Res.

Reset value

DTPE

Reset value
Res.

Reset value

Res.

Reset value

Res.

TIM15_BDTR

Res.

TIM15_CCR2

Res.

0x48 0x50
TIM15_CCR1

Res.

0x44
TIM15_RCR

Res.

0x38 0x40
TIM15_ARR

BKBID

0x38
TIM15_PSC

Res.

0x34
0

Res.

0x30
Reset value

Res.

0x2C

Res.

0x28
TIM15_CNT

Res.

0x24
TIM15_CCER

Res.

0x20
TIM15_CCMR1
Output
Compare mode

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

TIM15_CCMR1
Input Capture
mode

Res.

0x18
0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0
0
0

Reset value
0

CNT[15:0]

0
0

0
0

0
0

0

0
0
0
0

0

0

0

0

Reset value

0

0

0

0

0

0

CCR1[19:0]

CCR2[19:0]

Res.

DT[7:0]

0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0

0

0

0

0

0

0

0

CC1E

0

0

CC1P

0

0

CC1NE

0

0

CC2E

0

CC1NP

0

CC2P

0
OC1CE

OC2M
[2:0]

CC2S
[1:0]

Res.

0
IC2
PSC
[1:0]

Res.

IC2F[3:0]

CC2NP

0

0

0

0

OC1M
[2:0]

0

0

IC1
PSC
[1:0]
CC1S
[1:0]

0
0
0

OC1FE

IC1F[3:0]

OC1PE

31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0

Register
name

Res.

Offset

Res.

RM0481
General purpose timers (TIM15/TIM16/TIM17)

Table 446. TIM15 register map and reset values (continued)

CC1S
[1:0]

0
0
0

0

0

0

0

0
0
0
0
0
0

0
0
0
0
0
0

PSC[15:0]

ARR[19:0]

0
0
0
0
0
0
0
0

1
1
1
1
1
1
1
1

REP[7:0]

DTGF[7:0]

Res.

1847/3119

1874


Table 446. TIM15 register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x5C   | TIM15_TISEL   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x60   | TIM15_AF1     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x64   | TIM15_AF2     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | OCR           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | SEL[2:0]      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x68 - | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x3DC  | TIM15_DCR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DBSS[3:0]     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | DBL[4:0]      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | DBA[4:0]      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x3E0  | TIM15_DMAR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | DMAB[31:0]    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3 on page 111 for the register boundary addresses.
42.8 TIM16/TIM17 registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

42.8.1 TIMx control register 1 (TIMx_CR1) (x = 16 to 17)

Address offset: 0x00

Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 DITHEN: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 UIFREMAP: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 CKD[1:0]: Clock division

This bit-field indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (tDTS) used by the dead-time generators and the digital filters (tim_tix),

00: tDTS = tim_ker_ck
01: tDTS = 2×tim_ker_ck
10: tDTS = 4×tim_ker_ck
11: Reserved

Bit 7 ARPE: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 OPM: One pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 URS: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled.
These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
1: nly counter overflow/underflow generates an update interrupt or DMA request if enabled.
Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.
- 0: UEV enabled. The Update (UEV) event is generated by one of the following events:
  - Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.
- 1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
- 0: Counter disabled
- 1: Counter enabled

*Note:* External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

### 42.8.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

**Address offset:** 0x04

**Reset value:** 0x0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bits 15:10** Reserved, must be kept at reset value.

**Bit 9 OIS1N**: Output Idle state 1 (tim_oc1n output)
- 0: tim_oc1n=0 after a dead-time when MOE=0
- 1: tim_oc1n=1 after a dead-time when MOE=0

*Note:* This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

**Bit 8 OIS1**: Output Idle state 1 (tim_oc1 output)
- 0: tim_oc1=0 after a dead-time when MOE=0
- 1: tim_oc1=1 after a dead-time when MOE=0

*Note:* This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

**Bits 7:4** Reserved, must be kept at reset value.

**Bit 3 CCDS**: Capture/compare DMA selection
- 0: CCx DMA request sent when CCx event occurs
- 1: CCx DMA requests sent when update event occurs

**Bit 2 CCUS**: Capture/compare control update selection
- 0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.
- 1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when a rising edge occurs on tim_trgi (if available).

*Note:* This bit acts only on channels that have a complementary output.

**Bit 1** Reserved, must be kept at reset value.
Bit 0 **CCPC**: Capture/compare preloaded control
   0: CCxE, CCxNE and OCxM bits are not preloaded
   1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated
      only when COM bit is set.

   *Note: This bit acts only on channels that have a complementary output.*

### 42.8.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:10</td>
<td>Reserved</td>
<td>rw</td>
<td>0x0000</td>
</tr>
<tr>
<td>9</td>
<td>CC1DE: Capture/Compare 1 DMA request enable</td>
<td>rw</td>
<td>0x0000</td>
</tr>
<tr>
<td>8</td>
<td>UDE: Update DMA request enable</td>
<td>rw</td>
<td>0x0000</td>
</tr>
<tr>
<td>7</td>
<td>BIE: Break interrupt enable</td>
<td>rw</td>
<td>0x0000</td>
</tr>
<tr>
<td>6</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td>0x0000</td>
</tr>
<tr>
<td>5</td>
<td>COMIE: COM interrupt enable</td>
<td>rw</td>
<td>0x0000</td>
</tr>
<tr>
<td>4:2</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td>0x0000</td>
</tr>
<tr>
<td>1</td>
<td>CC1IE: Capture/Compare 1 interrupt enable</td>
<td>rw</td>
<td>0x0000</td>
</tr>
<tr>
<td>0</td>
<td>UIE: Update interrupt enable</td>
<td>rw</td>
<td>0x0000</td>
</tr>
</tbody>
</table>
42.8.4 TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 CC1OF: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.

0: No overcapture has been detected
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 BIF: Break interrupt flag
This flag is set by hardware as soon as the tim_brk input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred
1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 COMIF: COM interrupt flag
This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred
1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 CC1IF: Capture/Compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred
1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).
Bit 0  **UIF**: Update interrupt flag
   This bit is set by hardware on an update event. It is cleared by software.
   0: No update occurred.
   1: Update interrupt pending. This bit is set by hardware when the registers are updated:
      - At overflow regarding the repetition counter value (update if repetition counter = 0)
        and if the UDIS=0 in the TIMx_CR1 register.
      - When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if
        URS=0 and UDIS=0 in the TIMx_CR1 register.

### 42.8.5  TIMx event generation register (TIMx_EGR)(x = 16 to 17)

Address offset: 0x14
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

- **Bit 15:8** Reserved, must be kept at reset value.
- **Bit 7** **BG**: Break generation
   This bit is set by software in order to generate an event, it is automatically cleared by
   hardware.
   0: No action.
   1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or
   DMA transfer can occur if enabled.
- **Bit 6** Reserved, must be kept at reset value.
- **Bit 5** **COMG**: Capture/Compare control update generation
   This bit can be set by software, it is automatically cleared by hardware.
   0: No action
   1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits
   
   **Note**: This bit acts only on channels that have a complementary output.
- **Bits 4:2** Reserved, must be kept at reset value.
- **Bit 1** **CC1G**: Capture/Compare 1 generation
   This bit is set by software in order to generate an event, it is automatically cleared by
   hardware.
   0: No action.
   1: A capture/compare event is generated on channel 1:
   - **If channel CC1 is configured as output**:
     CC1IF flag is set. Corresponding interrupt or DMA request is sent if enabled.
   - **If channel CC1 is configured as input**:
     The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set,
     the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the
     CC1IF flag was already high.
- **Bit 0** **UG**: Update generation
   This bit can be set by software, it is automatically cleared by hardware.
   0: No action.
   1: Reinitialize the counter and generates an update of the registers. Note that the prescaler
   counter is cleared too (anyway the prescaler ratio is not affected).
42.8.6  **TIMx capture/compare mode register 1 (TIMx_CCMR1)**  
(x = 16 to 17)

Address offset: 0x18  
Reset value: 0x0000 0000  

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

### Input capture mode

<table>
<thead>
<tr>
<th>Bit 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7:4</td>
<td><strong>IC1F[3:0]</strong>: Input capture 1 filter</td>
</tr>
<tr>
<td></td>
<td>This bit-field defines the frequency used to sample tim_t1 input and the length of the digital filter applied to tim_t1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:</td>
</tr>
<tr>
<td></td>
<td>0000: No filter, sampling is done at fDTS</td>
</tr>
<tr>
<td></td>
<td>0001: fSAMPLING=fim_ker_ck, N=2</td>
</tr>
<tr>
<td></td>
<td>0010: fSAMPLING=fim_ker_ck, N=4</td>
</tr>
<tr>
<td></td>
<td>0011: fSAMPLING=fim_ker_ck, N=8</td>
</tr>
<tr>
<td></td>
<td>0100: fSAMPLING=fDTS/2, N=</td>
</tr>
<tr>
<td></td>
<td>0101: fSAMPLING=fDTS/2, N=8</td>
</tr>
<tr>
<td></td>
<td>0110: fSAMPLING=fDTS/4, N=6</td>
</tr>
<tr>
<td></td>
<td>0111: fSAMPLING=fDTS/4, N=8</td>
</tr>
<tr>
<td></td>
<td>1000: fSAMPLING=fDTS/8, N=6</td>
</tr>
<tr>
<td></td>
<td>1001: fSAMPLING=fDTS/8, N=8</td>
</tr>
<tr>
<td></td>
<td>1010: fSAMPLING=fDTS/16, N=5</td>
</tr>
<tr>
<td></td>
<td>1011: fSAMPLING=fDTS/16, N=6</td>
</tr>
<tr>
<td></td>
<td>1100: fSAMPLING=fDTS/16, N=8</td>
</tr>
<tr>
<td></td>
<td>1101: fSAMPLING=fDTS/32, N=5</td>
</tr>
<tr>
<td></td>
<td>1110: fSAMPLING=fDTS/32, N=6</td>
</tr>
<tr>
<td></td>
<td>1111: fSAMPLING=fDTS/32, N=8</td>
</tr>
</tbody>
</table>

| Bit 3:2  | **IC1PSC[1:0]**: Input capture 1 prescaler |
|          | This bit-field defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register). |
|          | 00: no prescaler, capture is done each time an edge is detected on the capture input. |
|          | 01: capture is done once every 2 events |
|          | 10: capture is done once every 4 events |
|          | 11: capture is done once every 8 events |
Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
- Others: Reserved

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).*

**42.8.7 TIMx capture/compare mode register 1 [alternate]**
**(TIMx_CCMR1)(x = 16 to 17)**

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OC1M</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[3]</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

**Output compare mode:**

- Bits 31:17 Reserved, must be kept at reset value.
- Bits 15:8 Reserved, must be kept at reset value.
- Bit 7 **OC1CE**: Output Compare 1 clear enable
  - 0: tim_oc1ref is not affected by the tim_ocref_clr input.
  - 1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr input.
Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal `tim_oc1ref` from which `tim_oc1` and `tim_oc1n` are derived. `tim_oc1ref` is active high whereas `tim_oc1` and `tim_oc1n` active level depends on `CC1P` and `CC1NP` bits.

- **0000**: Frozen - The comparison between the output compare register `TIMx_CCR1` and the counter `TIMx_CNT` has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.
- **0001**: Set channel 1 to active level on match. `tim_oc1ref` signal is forced high when the counter `TIMx_CNT` matches the capture/compare register 1 (`TIMx_CCR1`).
- **0010**: Set channel 1 to inactive level on match. `tim_oc1ref` signal is forced low when the counter `TIMx_CNT` matches the capture/compare register 1 (`TIMx_CCR1`).
- **0011**: Toggle - `tim_oc1ref` toggles when `TIMx_CNT=TIMx_CCR1`.
- **0100**: Force inactive level - `tim_oc1ref` is forced low.
- **0101**: Force active level - `tim_oc1ref` is forced high.
- **0110**: PWM mode 1 - Channel 1 is active as long as `TIMx_CNT<TIMx_CCR1` else inactive.
- **0111**: PWM mode 2 - Channel 1 is inactive as long as `TIMx_CNT<TIMx_CCR1` else active.
- Others: Reserved

**Note**: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in `TIMx_BDTR` register) and `CC1S='00'` (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output Compare 1 preload enable

- **0**: Preload register on `TIMx_CCR1` disabled. `TIMx_CCR1` can be written at anytime, the new value is taken in account immediately.
- **1**: Preload register on `TIMx_CCR1` enabled. Read/Write operations access the preload register. `TIMx_CCR1` preload value is loaded in the active register at each update event.

**Note**: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in `TIMx_BDTR` register) and `CC1S='00'` (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in `TIMx_CR1` register), to have the output pulse starting as soon as possible after the starting trigger.

- **0**: `CC1` behaves normally depending on counter and `CCR1` values even when the trigger is ON. The minimum delay to activate `CC1` output when an edge occurs on the trigger input is 5 clock cycles.
- **1**: An active edge on the trigger input acts like a compare match on `CC1` output. Then, `tim_ocx` is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate `CC1` output is reduced to 3 clock cycles.

**OC1FE** acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

- **00**: `CC1` channel is configured as output
- **01**: `CC1` channel is configured as input, `tim_ic1` is mapped on `tim_t1`
- Others: Reserved

**Note**: `CC1S` bits are writable only when the channel is OFF (`CC1E = ‘0’` in `TIMx_CCER`).
42.8.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity
- CC1 channel configured as output:
  - 0: tim_oc1n active high
  - 1: tim_oc1n active low
- CC1 channel configured as input:
  This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1. Refer to the description of CC1P.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).
On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.*
General purpose timers (TIM15/TIM16/TIM17)

Bit 2  **CC1NE**: Capture/Compare 1 complementary output enable
0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1  **CC1P**: Capture/Compare 1 output polarity
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

*When CC1 channel is configured as input*, both CC1NP/CC1P bits select the active polarity of T11FP1 and T12FP1 for trigger or capture operations.

- CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to T1xFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), T1xFP1 is not inverted (trigger operation in gated mode).
- CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to T1xFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), T1xFP1 is inverted (trigger operation in gated mode).
- CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both T1xFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), T1xFP1 is not inverted (trigger operation in gated mode).
- CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

*Note*: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0  **CC1E**: Capture/Compare 1 output enable
0: Capture mode disabled / OC1 is not active (see below)
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

*When CC1 channel is configured as output*, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 447* for details.
### Table 447. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)

<table>
<thead>
<tr>
<th>Control bits</th>
<th>MOE bit</th>
<th>OSSI bit</th>
<th>OSSR bit</th>
<th>CC1E bit</th>
<th>CC1NE bit</th>
<th>Output states(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>Output Disabled (not driven by the timer: Hi-Z)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>Output Disabled (not driven by the timer: Hi-Z)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>Output Disabled (not driven by the timer: Hi-Z)</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>Complementary to tim_oc1ref (not tim_oc1ref) + Polarity + dead-time</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>Off-State (output enabled with inactive state)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>Off-State (output enabled with inactive state)</td>
</tr>
</tbody>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OIS1, OIS1N, CC1P and CC1NP bits must be kept cleared.

**Note:** The state of the external I/O pins connected to the complementary tim_oc1 and tim_oc1n channels depends on the tim_oc1 and tim_oc1n channel state and GPIO control and alternate function selection registers.
42.8.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>UIFCPY: UIF Copy</td>
<td>r</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **UIFCPY: UIF Copy**
This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]: Counter value**

- **Non-dithering mode (DITHEN = 0)**
  - The register holds the counter value.
- **Dithering mode (DITHEN = 1)**
  - The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

42.8.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

Address offset: 0x28
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:0 **PSC[15:0]: Prescaler value**

The counter clock frequency (tim_cnt_ck) is equal to f_{tim_psc_ck} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).
42.8.11  TIMx auto-reload register (TIMx_ARR)(x = 16 to 17)

Address offset: 0x2C  
Reset value: 0x0000 FFFF

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19-0</td>
<td>ARR[19:0]: Auto-reload value</td>
</tr>
</tbody>
</table>

ARR is the value to be loaded in the actual auto-reload register. Refer to the Section 42.4.3: Time-base unit on page 1776 for more details about ARR update and behavior. The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)  
The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)  
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

42.8.12  TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30  
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>7-0</td>
<td>REP[7:0]: Repetition counter reload value</td>
</tr>
</tbody>
</table>

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enabled. It also defines the update interrupt generation rate, if this interrupt is enabled. When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

- the number of PWM periods in edge-aligned mode
- the number of half PWM period in center-aligned mode
### 42.8.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

Address offset: 0x34  
Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:20  Reserved, must be kept at reset value.  
Bits 19:0  CCR1[19:0]: Capture/Compare 1 value

**If channel CC1 is configured as output:**
- CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
- It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
- The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

**Non-dithering mode (DITHEN = 0)**
- The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

**Dithering mode (DITHEN = 1)**
- The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

**If channel CC1 is configured as input:**
- CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1).

**Non-dithering mode (DITHEN = 0)**
- The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

**Dithering mode (DITHEN = 1)**
- The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.
42.8.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44
Reset value: 0x0000 0000

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |   |   |   |   |   |   |   |   |   |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |   |   |   |   |   |   |   |   |   |

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional
0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode

In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm
0: Break input tim_brk is armed
1: Break input tim_brk is disarmed

This bit is cleared by hardware when no break source is active. The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
Bits 25:20  Reserved, must be kept at reset value.

Bits 19:16  **BKF[3:0]**: Break filter
This bit-field defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
- 0000: No filter, tim_brk acts asynchronously
- 0001: $f_{SAMPLING} = f_{tim.ker_ck}$, N=2
- 0010: $f_{SAMPLING} = f_{tim.ker_ck}$, N=4
- 0011: $f_{SAMPLING} = f_{tim.ker_ck}$, N=8
- 0100: $f_{SAMPLING} = f_{DTS}/2$, N=6
- 0101: $f_{SAMPLING} = f_{DTS}/2$, N=8
- 0110: $f_{SAMPLING} = f_{DTS}/4$, N=6
- 0111: $f_{SAMPLING} = f_{DTS}/4$, N=8
- 1000: $f_{SAMPLING} = f_{DTS}/8$, N=6
- 1001: $f_{SAMPLING} = f_{DTS}/8$, N=8
- 1010: $f_{SAMPLING} = f_{DTS}/16$, N=5
- 1011: $f_{SAMPLING} = f_{DTS}/16$, N=6
- 1100: $f_{SAMPLING} = f_{DTS}/32$, N=5
- 1101: $f_{SAMPLING} = f_{DTS}/32$, N=6
- 1110: $f_{SAMPLING} = f_{DTS}/32$, N=8
- 1111: $f_{SAMPLING} = f_{DTS}/32$, N=8

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15  **MOE**: Main output enable
This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.
- 0: tim_oc1 and tim_oc1n outputs are disabled or forced to idle state depending on the OSSI bit.
- 1: tim_oc1 and tim_oc1n outputs are enabled if their respective enable bits are set (CC1E, CC1NE in TIMx_CCER register)

See tim_oc1/tim_oc1n enable description for more details ([Section 42.8.8: TIMx capture/compare enable register (TIMx_CCER) (x = 16 to 17) on page 1857](#)).

Bit 14  **AOE**: Automatic output enable
- 0: MOE can be set only by software
- 1: MOE can be set by software or automatically at the next update event (if the tim_brk input is not active)

**Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13  **BKP**: Break polarity
- 0: Break input tim_brk is active low
- 1: Break input tim_brk is active high

**Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12  **BKE**: Break enable
- 0: Break inputs (tim_brk and tim_sys_brk event) disabled
- 1: Break inputs (tim_brk and tim_sys_brk event) enabled

**Note:** This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_oc1/tim_oc1n enable description for more details (Section 42.8.8: TIMx capture/compare enable register (TIMx_CCER) (x = 16 to 17) on page 1857).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_oc1/tim_oc1n outputs are enabled with their inactive level as soon as CC1E=1 or CC1NE=1 (the output is still controlled by the timer).

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See tim_oc1/tim_oc1n enable description for more details (Section 42.8.8: TIMx capture/compare enable register (TIMx_CCER) (x = 16 to 17) on page 1857).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (tim_oc1/tim_oc1n enable output signal=0)

1: When inactive, tim_oc1/tim_oc1n outputs are forced first with their idle level as soon as CC1E=1 or CC1NE=1. tim_oc1/tim_oc1n enable output signal=1)

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).*

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

*Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.*

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

\[ DTG[7:5]=0xx \Rightarrow DT=DTG[7:0] \times t_{\text{dts}} \text{ with } t_{\text{dts}}=t_{\text{DTS}} \]

\[ DTG[7:5]=10x \Rightarrow DT=(64+DTG[5:0]) \times t_{\text{dts}} \text{ with } T_{\text{dts}}=2x_{\text{DTS}} \]

\[ DTG[7:5]=110 \Rightarrow DT=(32+DTG[4:0]) \times t_{\text{dts}} \text{ with } T_{\text{dts}}=8x_{\text{DTS}} \]

\[ DTG[7:5]=111 \Rightarrow DT=(32+DTG[4:0]) \times t_{\text{dts}} \text{ with } T_{\text{dts}}=16x_{\text{DTS}} \]

Example if T_{\text{DTS}}=125ns (8MHz), dead-time possible values are:

- 0 to 15875 ns by 125 ns steps,
- 16 µs to 31750 ns by 250 ns steps,
- 32 µs to 63 µs by 1 µs steps,
- 64 µs to 126 µs by 2 µs steps

*Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*
### 42.8.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17)

Address offset: 0x054  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:18  Reserved, must be kept at reset value.

**Bit 17 DTPE**: Deadtime preload enable
- 0: Deadtime value is not preloaded
- 1: Deadtime value preload is enabled

*Note*: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

**Bit 16 DTAE**: Deadtime asymmetric enable
- 0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register
- 1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

*Note*: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:8  Reserved, must be kept at reset value.

**Bits 7:0 DTGF[7:0]**: Dead-time falling edge generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

- \( \text{DTGF}[7:5]=0xx \rightarrow \text{DTF} = \text{DTGF}[7:0] \times t_{\text{dtg}} = t_{\text{DTS}} \)
- \( \text{DTGF}[7:5]=10x \rightarrow \text{DTF} = (64+\text{DTGF}[5:0]) \times t_{\text{dtg}} = 2x t_{\text{DTS}} \)
- \( \text{DTGF}[7:5]=110 \rightarrow \text{DTF} = (32+\text{DTGF}[4:0]) \times t_{\text{dtg}} = 8x t_{\text{DTS}} \)
- \( \text{DTGF}[7:5]=111 \rightarrow \text{DTF} = (32+\text{DTGF}[4:0]) \times t_{\text{dtg}} = 16x t_{\text{DTS}} \)

Example if \( T_{\text{DTS}}=125\text{ns} \) (8MHz), dead-time possible values are:
- 0 to 15875 ns by 125 ns steps,
- 16 us to 31750 ns by 250 ns steps,
- 32 us to 63us by 1 us steps,
- 64 us to 126 us by 2 us steps

*Note*: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).
42.8.16  **TIMx input selection register (TIMx_TISEL)(x = 16 to 17)**

Address offset: 0x5C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **TI1SEL[3:0]**: selects tim_t1_in[15:0] input

- 0000: TIMx_CH1 input (tim_t1_in0)
- 0001: tim_t1_in1
- ...
- 1111: tim_t1_in15

Refer to **Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals** for interconnects list.

42.8.17  **TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17)**

Address offset: 0x060  
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:14  Reserved, must be kept at reset value.

Bit 13  **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

- 0: tim_brk_cmp4 input is active high
- 1: tim_brk_cmp4 input is active low

**Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity
This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the
BKP polarity bit.
0: tim_brk_cmp3 input is active high
1: tim_brk_cmp3 input is active low

*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity
This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the
BKP polarity bit.
0: tim_brk_cmp2 input is active high
1: tim_brk_cmp2 input is active low

*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity
This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the
BKP polarity bit.
0: tim_brk_cmp1 input is active high
1: tim_brk_cmp1 input is active low

*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity
This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed
together with the BKP polarity bit.
0: TIMx_BKIN input is active high
1: TIMx_BKIN input is active low

*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable
This bit enables the tim_brk_cmp8 for the timer’s tim_brk input. mdf_brkx output is 'ORed'
with the other tim_brk sources.
0: tim_brk_cmp8 input disabled
1: tim_brk_cmp8 input enabled

*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable
This bit enables the tim_brk_cmp7 for the timer’s tim_brk input. tim_brk_cmp7 output is
'ORed' with the other tim_brk sources.
0: tim_brk_cmp7 input disabled
1: tim_brk_cmp7 input enabled

*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable
This bit enables the tim_brk_cmp6 for the timer’s tim_brk input. tim_brk_cmp6 output is
'ORed' with the other tim_brk sources.
0: tim_brk_cmp6 input disabled
1: tim_brk_cmp6 input enabled

*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 5 **BKCMP5E**: tim\_brk\_cmp5 enable

This bit enables the tim\_brk\_cmp5 for the timer’s tim\_brk input. tim\_brk\_cmp5 output is ‘OREd’ with the other tim\_brk sources.

- 0: tim\_brk\_cmp5 input disabled
- 1: tim\_brk\_cmp5 input enabled

**Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 4 **BKCMP4E**: tim\_brk\_cmp4 enable

This bit enables the tim\_brk\_cmp4 for the timer’s tim\_brk input. tim\_brk\_cmp4 output is ‘OREd’ with the other tim\_brk sources.

- 0: tim\_brk\_cmp4 input disabled
- 1: tim\_brk\_cmp4 input enabled

**Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 3 **BKCMP3E**: tim\_brk\_cmp3 enable

This bit enables the tim\_brk\_cmp3 for the timer’s tim\_brk input. tim\_brk\_cmp3 output is ‘OREd’ with the other tim\_brk sources.

- 0: tim\_brk\_cmp3 input disabled
- 1: tim\_brk\_cmp3 input enabled

**Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 2 **BKCMP2E**: tim\_brk\_cmp2 enable

This bit enables the tim\_brk\_cmp2 for the timer’s tim\_brk input. tim\_brk\_cmp2 output is ‘OREd’ with the other tim\_brk sources.

- 0: tim\_brk\_cmp2 input disabled
- 1: tim\_brk\_cmp2 input enabled

**Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 1 **BKCMP1E**: tim\_brk\_cmp1 enable

This bit enables the tim\_brk\_cmp1 for the timer’s tim\_brk input. tim\_brk\_cmp1 output is ‘OREd’ with the other tim\_brk sources.

- 0: tim\_brk\_cmp1 input disabled
- 1: tim\_brk\_cmp1 input enabled

**Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 0 **BKINE**: TIMx\_BKIN input enable

This bit enables the TIMx\_BKIN alternate function input for the timer’s tim\_brk input. TIMx\_BKIN input is ‘OREd’ with the other tim\_brk sources.

- 0: TIMx\_BKIN input disabled
- 1: TIMx\_BKIN input enabled

**Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).
### 42.8.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17)

**Address offset**: 0x064  
**Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Format</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x064</td>
<td>rw</td>
<td>OCRSEL[2:0]</td>
<td>tim_ocref_clr source selection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>000: tim_ocref_clr0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>001: tim_ocref_clr1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>010: tim_ocref_clr2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>011: tim_ocref_clr3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>100: tim_ocref_clr4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>101: tim_ocref_clr5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>110: tim_ocref_clr6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>111: tim_ocref_clr7</td>
</tr>
</tbody>
</table>

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: tim_ocref_clr source selection

These bits select the tim_ocref_clr input source.

- 000: tim_ocref_clr0
- 001: tim_ocref_clr1
- 010: tim_ocref_clr2
- 011: tim_ocref_clr3
- 100: tim_ocref_clr4
- 101: tim_ocref_clr5
- 110: tim_ocref_clr6
- 111: tim_ocref_clr7

Refer to *Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

**Note**: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

### 42.8.19 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

**Address offset**: 0x03DC  
**Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Format</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x03DC</td>
<td>rw</td>
<td>DBSS[3:0]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Format</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x03DC</td>
<td></td>
<td>DBL[4:0]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Format</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x03DC</td>
<td></td>
<td>DBA[4:0]</td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:16  **DBSS[3:0]: DMA burst source selection**
This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).
- 0000: Reserved
- 0001: Update
- 0010: CC1
- Other: reserved

Bits 15:13  Reserved, must be kept at reset value.

Bits 12:8  **DBL[4:0]: DMA burst length**
This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).
- 00000: 1 transfer,
- 00001: 2 transfers,
- 00010: 3 transfers,
- ... 10001: 18 transfers.

Bits 7:5  Reserved, must be kept at reset value.

Bits 4:0  **DBA[4:0]: DMA base address**
This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

**Example:**
- 00000: TIMx_CR1,
- 00001: TIMx_CR2,
- 00010: TIMx_SMCR,
- ...

**Example:** Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

### 42.8.20 TIM16/TIM17 DMA address for full transfer
(TIMx_DMAR) (x = 16 to 17)

Address offset: 0x3E0

Reset value: 0x0000 0000
Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

\[(\text{TIM}x\_\text{CR1 address}) + (\text{DBA} + \text{DMA index}) \times 4\]

where TIM\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM\_DCR).
### 42.8.21 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>TIMx_CR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>TIMx_CR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>TIMx_DIER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>TIMx_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>TIMx_EGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>TIMx_CCMR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Input Capture</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output Compare</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>TIMx_CCER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td>TIMx_CNT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x28</td>
<td>TIMx_PSC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 448. TIM16/TIM17 register map and reset values**
Refer to Section 2.3 for the register boundary addresses.
43 Low-power timer (LPTIM)

43.1 Introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. Also, the LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

43.2 LPTIM main features

- 16 bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no LP oscillator running, used by Pulse Counter application)
- 16 bit ARR autoreload register
- 16 bit capture/compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable Digital Glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode
- Repetition counter
- Up to 2 independent channels for:
  - Input capture
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on 10 events
- DMA request generation on the following events:
  - Update event
  - Input capture
43.3 LPTIM implementation

*Table 449* describes LPTIM implementation on STM32H563/H573 and STM32H562 devices. The full set of features is implemented in LPTIM1, 2, 3, 5 and 6. LPTIM4 supports a smaller set of features.

<table>
<thead>
<tr>
<th>LPTIM modes/features(1)</th>
<th>LPTIM1</th>
<th>LPTIM2</th>
<th>LPTIM3</th>
<th>LPTIM4</th>
<th>LPTIM5</th>
<th>LPTIM6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PWM mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Input Capture</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Number of channels</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Number of DMA requests</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Wake-up from Stop mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Autonomous mode</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1. X = supported.
43.4 LPTIM functional description

43.4.1 LPTIM block diagram

Figure 595. LPTIM1/2/3/5/6 timer block diagram

1. Some IOs may not be available, refer to Section 43.4.2: LPTIM pins and internal signals.
1. Some IOs may not be available, refer to Section 43.4.2: LPTIM pins and internal signals.

### 43.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

#### Table 450. LPTIM1/2/3/5/6 input/output pins

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM_IN1</td>
<td>Digital input</td>
<td>LPTIM Input 1 from GPIO pin on mux input 0</td>
</tr>
<tr>
<td>LPTIM_IN2</td>
<td>Digital input</td>
<td>LPTIM Input 2 from GPIO pin on mux input 0</td>
</tr>
<tr>
<td>LPTIM_ETR</td>
<td>Digital input</td>
<td>LPTIM external trigger GPIO pin</td>
</tr>
<tr>
<td>LPTIM_CH1</td>
<td>Digital input/output</td>
<td>LPTIM channel 1 Input/Output GPIO pin</td>
</tr>
<tr>
<td>LPTIM_CH2</td>
<td>Digital input/output</td>
<td>LPTIM channel 2 Input/Output GPIO pin</td>
</tr>
</tbody>
</table>

#### Table 451. LPTIM4 input/output pins

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM_IN1</td>
<td>Digital input</td>
<td>LPTIM Input 1 from GPIO pin on mux input 0</td>
</tr>
<tr>
<td>LPTIM_ETR</td>
<td>Digital input</td>
<td>LPTIM external trigger GPIO pin</td>
</tr>
<tr>
<td>LPTIM_OUT</td>
<td>Digital output</td>
<td>LPTIM Output GPIO pin</td>
</tr>
</tbody>
</table>
### Table 452. LPTIM1/2/3/5/6 internal signals

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_pclk</td>
<td>Digital input</td>
<td>LPTIM APB clock domain</td>
</tr>
<tr>
<td>lptim_ker_ck</td>
<td>Digital input</td>
<td>LPTIM kernel clock</td>
</tr>
<tr>
<td>lptim_in1_mux1</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 1</td>
</tr>
<tr>
<td>lptim_in1_mux2</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 2</td>
</tr>
<tr>
<td>lptim_in1_mux3</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 3</td>
</tr>
<tr>
<td>lptim_in2_mux1</td>
<td>Digital input</td>
<td>Internal LPTIM input 2 connected to mux input 1</td>
</tr>
<tr>
<td>lptim_in2_mux2</td>
<td>Digital input</td>
<td>Internal LPTIM input 2 connected to mux input 2</td>
</tr>
<tr>
<td>lptim_in2_mux3</td>
<td>Digital input</td>
<td>Internal LPTIM input 2 connected to mux input 3</td>
</tr>
<tr>
<td>lptim_ic1_mux1</td>
<td>Digital input</td>
<td>Internal LPTIM input capture 1 connected to mux input 1</td>
</tr>
<tr>
<td>lptim_ic1_mux2</td>
<td>Digital input</td>
<td>Internal LPTIM input capture 1 connected to mux input 2</td>
</tr>
<tr>
<td>lptim_ic1_mux3</td>
<td>Digital input</td>
<td>Internal LPTIM input capture 1 connected to mux input 3</td>
</tr>
<tr>
<td>lptim_ic2_mux1</td>
<td>Digital input</td>
<td>Internal LPTIM input capture 2 connected to mux input 1</td>
</tr>
<tr>
<td>lptim_ic2_mux2</td>
<td>Digital input</td>
<td>Internal LPTIM input capture 2 connected to mux input 2</td>
</tr>
<tr>
<td>lptim_ic2_mux3</td>
<td>Digital input</td>
<td>Internal LPTIM input capture 2 connected to mux input 3</td>
</tr>
<tr>
<td>lptim_ext_trigx</td>
<td>Digital input</td>
<td>LPTIM external trigger input x</td>
</tr>
<tr>
<td>lptim_it</td>
<td>Digital output</td>
<td>LPTIM global interrupt</td>
</tr>
<tr>
<td>lptim_wakeup</td>
<td>Digital output</td>
<td>LPTIM wake-up event</td>
</tr>
<tr>
<td>lptim_ic1_dma</td>
<td>Digital output</td>
<td>LPTIM input capture 1 DMA request</td>
</tr>
<tr>
<td>lptim_ic2_dma</td>
<td>Digital output</td>
<td>LPTIM input capture 2 DMA request</td>
</tr>
<tr>
<td>lptim_ue_dma</td>
<td>Digital output</td>
<td>LPTIM update event DMA request</td>
</tr>
</tbody>
</table>

### Table 453. LPTIM4 internal signals

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_pclk</td>
<td>Digital input</td>
<td>LPTIM APB clock domain</td>
</tr>
<tr>
<td>lptim_ker_ck</td>
<td>Digital input</td>
<td>LPTIM kernel clock</td>
</tr>
<tr>
<td>lptim_in1_mux1</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 1</td>
</tr>
<tr>
<td>lptim_in1_mux2</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 2</td>
</tr>
<tr>
<td>lptim_in1_mux3</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 3</td>
</tr>
<tr>
<td>lptim_ext_trigx</td>
<td>Digital input</td>
<td>LPTIM external trigger input x</td>
</tr>
<tr>
<td>lptim_out</td>
<td>Digital output</td>
<td>LPTIM counter output</td>
</tr>
<tr>
<td>lptim_it</td>
<td>Digital output</td>
<td>LPTIM global interrupt</td>
</tr>
<tr>
<td>lptim_wakeup</td>
<td>Digital output</td>
<td>LPTIM wake-up event</td>
</tr>
</tbody>
</table>
43.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter.

**Table 454. LPTIM1/2/3/4/5/6 external trigger connection**

<table>
<thead>
<tr>
<th>TRIGSEL</th>
<th>External trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM1</td>
<td>LPTIM2</td>
</tr>
<tr>
<td>lptim_ext_trig0</td>
<td>GPIO</td>
</tr>
<tr>
<td>lptim_ext_trig1</td>
<td>rtc_alra_trg</td>
</tr>
<tr>
<td>lptim_ext_trig2</td>
<td>rtc_alrb_trg</td>
</tr>
<tr>
<td>lptim_ext_trig3</td>
<td>tamp_trg1</td>
</tr>
<tr>
<td>lptim_ext_trig4</td>
<td>tamp_trg2</td>
</tr>
<tr>
<td>lptim_ext_trig5</td>
<td>gpdma_ch1_tcf</td>
</tr>
<tr>
<td>lptim_ext_trig6</td>
<td>-</td>
</tr>
<tr>
<td>lptim_ext_trig7</td>
<td>-</td>
</tr>
</tbody>
</table>

**Table 455. LPTIM1/2/3/5/6 input 1 connection**

<table>
<thead>
<tr>
<th>LPTIM1/2/3/4 input 1 connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_in1_mux0</td>
</tr>
<tr>
<td>lptim_in1_mux1</td>
</tr>
<tr>
<td>lptim_in1_mux2</td>
</tr>
<tr>
<td>lptim_in1_mux3</td>
</tr>
</tbody>
</table>

**Table 456. LPTIM1/2/3/5/6 input 2 connection**

<table>
<thead>
<tr>
<th>LPTIM1/2/2 input 2 connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_in2_mux0</td>
</tr>
<tr>
<td>lptim_in2_mux1</td>
</tr>
<tr>
<td>lptim_in2_mux2</td>
</tr>
<tr>
<td>lptim_in2_mux3</td>
</tr>
</tbody>
</table>

**Table 457. LPTIM1/2/3/5/6 input capture 1 connection**

<table>
<thead>
<tr>
<th>LPTIM1/3 input capture 1 connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_ic1_mux0</td>
</tr>
<tr>
<td>lptim_ic1_mux1</td>
</tr>
<tr>
<td>lptim_ic1_mux2</td>
</tr>
<tr>
<td>lptim_ic1_mux3</td>
</tr>
</tbody>
</table>
43.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM may run in one of these two possible configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or Pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM uses an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal must also be provided (first configuration). In this case, the internal clock signal frequency must be at least four times higher than the external clock signal frequency.
43.4.5 **Glitch filter**

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source must first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

The digital filters are divided into three groups:

- The first group of digital filters protects the LPTIM internal or external inputs. The digital filters sensitivity is controlled by the CKFLT bits.
- The second group of digital filters protects the LPTIM internal or external trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.
- The third group of digital filters protects the LPTIM internal or external input captures. The digital filters sensitivity is controlled by the ICxF bits.

*Note:* The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that is detected on one of the LPTIM inputs to consider a signal level change as a valid transition. **Figure 597** shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

![Glitch filter timing diagram](MS32490V1)

*Note:* In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT, ICxF and TRGFLT bits to ‘0’. In that case, an external analog filter may be used to protect the LPTIM external inputs against glitches.

43.4.6 **Prescaler**

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] 3-bit field. The table below lists all the possible division ratios:

<table>
<thead>
<tr>
<th>programming</th>
<th>dividing factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>/1</td>
</tr>
<tr>
<td>001</td>
<td>/2</td>
</tr>
</tbody>
</table>

*Table 461. Prescaler division ratios*
43.4.7 Trigger multiplexer

The LPTIM counter may be started either by software or after the detection of an active edge on one of the 8 trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals ‘00’, The LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than ‘00’, TRIGSEL[2:0] is used to select which of the 8 trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. So after a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it is ignored (unless timeout function is enabled).

**Note:** The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled is discarded by hardware.

**Note:** When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

43.4.8 Operating mode

The LPTIM features two operating modes:

- **The Continuous mode:** the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- **One-shot mode:** the timer is started from a trigger event and stops when an LPTIM update event is generated.

**One-shot mode**

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event re-starts the timer. Any trigger event occurring after the counter starts and before the next LPTIM update event, is discarded.
In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the repetition counter has stopped (after the update event), and if the repetition register content is different from zero, the repetition counter gets reloaded with the value already contained by the repetition register and a new one-shot counting cycle is started as shown in Figure 598.

**Figure 598. LPTIM output waveform, single counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)**

- Set-once mode activated:

Note that when the WAVE bitfield in the LPTIM_CFRG register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in Figure 599.

**Figure 599. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set)**

In case of software start (TRIGEN[1:0] = ‘00’), the SNGSTRT setting starts the counter for one-shot counting.

**Continuous mode**

To enable the continuous counting, the CNTSTRT bit must be set.
In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set, starts the counter for continuous counting. Any subsequent external trigger event is discarded as shown in Figure 600.

In case of software start (TRIGEN[1:0] = ‘00’), setting CNTSTRT starts the counter for continuous counting.

**Figure 600. LPTIM output waveform, Continuous counting mode configuration**

SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (The ENABLE bit is set to ‘1’). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT switches the LPTIM to the One-shot mode. The counter (if active) stops as soon as an LPTIM update event is generated.

If the One-shot mode was previously selected, setting CNTSTRT switches the LPTIM to the Continuous mode. The counter (if active) restarts as soon as it reaches ARR.

### 43.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event starts the timer, any successive trigger event resets the LPTIM counter and the repetition counter and the timer restarts.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

### 43.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CCRx (capture/compare register), are used to generate several different waveforms on LPTIM output.

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CCRx. The LPTIM output is reset as soon as a
match occurs between the LPTIM_ARR and the LPTIM_CNT register. For more details see Section 43.4.19: PWM mode.

- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require that the LPTIM_ARR register value be strictly greater than the LPTIM_CCRx register value.

The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to ‘0’ forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to ‘1’ forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL/CCxP bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value changes immediately after the polarity is reconfigured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by 2 can be generated. Figure 601 below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL/CCxP bit.

**Figure 601. Waveform generation**
43.4.11 Register update

The LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register are updated immediately after the APB bus write operation or in synchronization with the next LPTIM update event if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated:

- When the PRELOAD bit is reset to ‘0’, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are immediately updated after any write access.
- When the PRELOAD bit is set to ‘1’, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated at next LPTIM update event, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag, the REPOK flag and the CMPxOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register.

After a write to the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag, the REPOK flag or the CMPxOK flag be set, leads to unpredictable results.

43.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM Input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source is used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0
    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1
    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.
    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal must never exceed the frequency of the internal clock.
provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRES[2:0] = 000).

- CKSEL = 1: the LPTIM is clocked by an external clock source
  COUNTMODE value is don’t care.
  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external Input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.
  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.
  Since the signal injected on the LPTIM external Input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

43.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock is needed before the LPTIM is actually enabled.

The LPTIM_CFGR register must be modified only when the LPTIM is disabled.

43.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register to zero, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bitfield to ‘1’, the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic elapse before the reset is taken into account. This makes the LPTIM counter count few extra pluses between the time when the reset is trigger and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing ‘1’ to the COUNTRST bit.

- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to ‘1’, any read access to the LPTIM_CNT register resets its content to zero. Asynchronous reset must be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset must be applied only when there is enough insurance that no toggle occurs on the LPTIM Input1.
  Note that to read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. So
43.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two Down and Up flags in the LPTIM_ISR register. Also, an interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to ‘1’. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder’s position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

<table>
<thead>
<tr>
<th>Active edge</th>
<th>Level on opposite signal (Input1 for Input2, Input2 for Input1)</th>
<th>Input1 signal</th>
<th>Input2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input1 signal</td>
<td>Rising</td>
<td>Falling</td>
</tr>
<tr>
<td>Rising Edge</td>
<td>High</td>
<td>Down</td>
<td>No count</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Up</td>
<td>No count</td>
</tr>
<tr>
<td>Falling Edge</td>
<td>High</td>
<td>No count</td>
<td>Up</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>No count</td>
<td>Down</td>
</tr>
<tr>
<td>Both Edges</td>
<td>High</td>
<td>Down</td>
<td>Up</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Up</td>
<td>Down</td>
</tr>
</tbody>
</table>

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.
Caution: In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to ‘0’. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be ‘000’).

Figure 602. Encoder mode counting sequence

43.4.16 Repetition Counter

The LPTIM features a repetition counter that decrements by 1 each time an LPTIM counter overflow event occurs. A repetition counter underflow event is generated when the repetition counter contains zero and the LPTIM counter overflows. Next to each repetition counter underflow event, the repetition counter gets loaded with the content of the REP[7:0] bitfield which belongs to the repetition register LPTIM_RCR.

A repetition underflow event is generated on each and every LPTIM counter overflow when the REP[7:0] register is set to 0.

When PRELOAD = 1, writing to the REP[7:0] bitfield has no effect on the content of the repetition counter until the next repetition underflow event occurs. The repetition counter continues to decrement each LPTIM counter overflow event and only when a repetition underflow event is generated, the new value written into REP[7:0] is loaded into the repetition counter. This behavior is depicted in Figure 603.
A repetition counter underflow event is systematically associated with LPTIM preloaded registers update (refer to section "Register update" for more information).

Repetition counter underflow event is signaled to the software through the update event (UE) flag mapped into the LPTIM_ISR register. When set, the UE flag can trigger an LPTIM interrupt if its respective update event interrupt enable (UEIE) control bit, mapped to the LPTIM_DIER register, is set.

The repetition register LPTIM_RCR is located in the APB bus interface clock domain where the repetition counter itself is located in the LPTIM kernel clock domain. Each time a new value is written to the LPTIM_RCR register, that new content is propagated from the APB bus interface clock domain to the LPTIM kernel clock domain so that the new written value is loaded to the repetition counter immediately after a repetition counter underflow event. The synchronization delay for the new written content is four APB clock cycles plus three LPTIM kernel clock cycles and it is signaled by the REPOK flag located in the LPTIM_ISR register when it is elapsed. When the LPTIM kernel clock cycle is relatively slow, for instance when the LPTIM kernel is being clocked by the LSI clock source, it can be lengthy to keep polling on the REPOK flag by software to detect that the synchronization of the LPTIM_RCR register content is finished. For that reason, the REPOK flag, when set, can generate an interrupt if its associated REPOKIE control bit in the LPTIM_DIER register is set.

**Note:** After a write to the LPTIM_RCR register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the REPOK flag is set, leads to unpredictable results.

**Caution:** When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the autoreload match event, otherwise an unpredictable behavior may occur.
43.4.17 Capture/compare channels

Each capture/compare channel is built around a capture/compare register, an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control) for PWM.

**Input stage**

The input stage samples the corresponding LPTIx input to generate a filtered signal LPTIxF. Then, an edge detector with polarity selection generates ICx signal used as the capture command. It is prescaled to generate the capture command signal (ICxPS).

**Output stage**

The output stage generates an intermediate waveform which is then used for reference: OCxREF (active high). The polarity acts at the end of the chain.

43.4.18 Input capture mode

In Input capture mode, the capture/compare registers (LPTIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. Assuming input capture is enabled on a channel x (CCxE set) and when a capture occurs, the corresponding CCxIF flag (LPTIM_ISR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (LPTIM_ISR register) is set. CCxIF can be cleared by software by writing the CCxICF to 1 or by reading the captured data stored in the LPTIM_CCRx register. CCxOF is cleared by writing CCxOCF to 1.

*Note:* In DMA mode, the input capture channel have to be enabled (set CCxE bit) the last, after enabling the IC DMA request and after starting the counter. This is in order to prevent generating an input capture DMA request when the counter is not started yet.

**Input capture Glitch filter latency**

When a trigger event arrives on channel x input (LPTIx) and depending on the configured glitch filter (ICxF[1:0] field in CCMRx register) and on the kernel clock prescaler value...
(PRESCH[2:0] field in CFGR register), there is a variable latency that leads to a systematic offset (see Table 463) between the captured value stored in the CCRx register and the real value corresponding to the capture trigger.

This offset has no impact on pulse width measurement as it is systematic and compensated between two captures.

The real capture value corresponding to the input capture trigger can be calculated using the below formula:

Real capture value = captured(LPTIM_CCRx) - offset

The relevant offset must be used depending on the glitch filter and on the kernel clock prescaler value (PRESCH field in CFGR register)

**Example:** determining the real capture value when PRESCH[2:0] = 0x2 and ICxF = 0x3.

For this configuration (PRESCH[2:0] = 0x2 and ICxF = 0x3) and according to the Table 463, the offset is 5.

Assuming that the captured value in CCRx is 9 (LPTIM_CNT = 9), this means that the capture trigger occurred when the LPTIM_CNT was equal to 9 - 5 = 4.

### Table 463. Input capture Glitch filter latency (in counter step unit)

<table>
<thead>
<tr>
<th>Prescaler PRESCH[2:0]</th>
<th>ICxF[1:0]</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
43.4.19 PWM mode

The PWM mode enables to generate a signal with a frequency determined by the value of the LPTIM_ARR register and a duty cycle determined by the value of the LPTIM_CCRx register. The LPTIM is able to generate PWM in edge-aligned mode.

OCx polarity is software programmable using the CCxP bit in the LPTIM_CCMRx register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the LPTIM_CCMRx register. Refer to the LPTIM_CCMRx register description for more details.

*Figure 606* gives an example where the LPTIM channel 1 is configured in PWM mode with LPTIM_CCR1 = 6 then 1 and LPTIM_ARR=10.

<table>
<thead>
<tr>
<th>Prescaler PRES[2:0]</th>
<th>ICxF[1:0]</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

*Figure 606. Edge-aligned PWM mode (PRELOAD = 1)*

In the following example the reference PWM signal OCxREF is low as long as LPTIM_CNT ≤ LPTIM_CCRx else it becomes high.
Figure 607 shows some edge-aligned PWM waveforms in an example where LPTIM.ARR = 8.

Figure 607. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0)

<table>
<thead>
<tr>
<th>Counter register</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCxREF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx=3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCxREF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx=6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCxREF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCxREF &quot;0&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PWM mode with immediate update PRELOAD = 0**

The PWM mode with PRELOAD = 0 enables the early change of the output level within the current PWM cycle. Based on the immediate update (PRELOAD = 0) of the LPTIM.CCRx register and on the continuous comparison of LPTIM.CNT and LPTIM.CCRx registers, it permits to have a new duty cycle value applied as soon as possible within the current PWM cycle, without having to wait for the completion of the current PWM period.

When the (PRELOAD = 0), the OCxREF signal level can be changed on-the-fly by software (or DMA) by updating the compare value in the LPTIM.CCRx register.

Depending on the written compare value and on the current counter and compare values, the OCxREF level is re-assigned as illustrated below:

- If the new compare value does not exceed the current counter value and the current compare value exceeds the counter, OCxREF level is re-assigned high as soon as the new compare value is written.
- If the new compare value exceeds the counter value and the current compare value does not exceed the counter, OCxREF level is re-assigned low as soon as the new compare value is written.

The output reference signal OCxREF level is left unchanged when none of the new compare value and the current compare value exceed the counter. Figure 608 illustrates the behavior of the OCxREF signal level when PRELOAD = 0 and PRELOAD = 1.
Note: For both PWM modes, the compare match, auto-reload match and the update event flags are set one LPTIM counter cycle later after the corresponding event, the OCxREF level is also changed one LPTIM counter cycle later after the corresponding event. For instance when the LPTIM_CCRx is set to 3 the CCxIF is set when the LPTIM_CNT = 4. Figure 606 illustrates this behavior.

43.4.20 DMA requests

The LPTIM has the capability to generate two categories of DMA requests:
- DMA requests used to retrieve the input-capture counter values
- DMA update requests are used to re-program part of the LPTIMER, multiple times, at regular intervals, without software overhead.

Input capture DMA request

Each LPTIM channel has its dedicated input capture DMA request. A DMA request is generated (if CCxDE bit is set in LPTIM_DIER) and CCxIF is set each time a capture is ready in the CCRx register. The captured values in CCRx can then be transferred regularly by DMA to the desired memory destination. The CCxIF is automatically cleared by hardware when the captured value in CCRx register is read.

Note: The ICx DMA request signal lptim_icx_dma is reset in the following conditions:
- if the corresponding DMA request is disabled (clear CCxDE bit in the LPTIM_DIER register)
- or if the channel x is disabled (clear CCxE bit)
- or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)

Update event DMA request

A DMA request is generated (if UEDE is set in LPTIM_DIER) and the UE flag is set at each update event. DMA request can be used to regularly update the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx registers permitting to generate custom PWM waveforms.
The UE is automatically cleared by hardware upon any bus master (like CPU or DMA) write access to the LPTIM_ARR register.

Note: The UE DMA request signal lptim_ue_dma is reset in the following conditions:
- if the corresponding DMA request is disabled (clear UEDE bit in the LPTIM DIER register)
- or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)

43.4.21 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the DBG_LPTIM_STOP configuration bit in the DBG module.

43.5 LPTIM low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. LPTIM interrupts cause the device to exit Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

Note: All DMA requests must be disabled (reset UEDE and CCxDE bits) before entering Sleep, Stop and Standby modes.

43.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_DIER register:
- Compare match
- Auto-reload match (whatever the direction if encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (encoder mode), programmable (up / down / both).
- Update Event
- Repetition register update OK
- Input capture occurred
- Over-capture occurred
- Interrupt enable register update OK

Note: If any bit in the LPTIM_DIER register is set after that its corresponding flag in the LPTIM_ISR register (Status Register) is set, the interrupt is not asserted.
Table 465. Interrupt events

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop mode(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIMx</td>
<td>Compare match</td>
<td>CCxIF</td>
<td>CCxIE</td>
<td>Write 1 to CCxCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Input capture(2)</td>
<td>CCxIF</td>
<td>CCxIE</td>
<td>Write 1 to CCxCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Over-capture(2)</td>
<td>CCxOF</td>
<td>CCxOE</td>
<td>Write 1 to CCxOCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Auto-reload match</td>
<td>ARRM</td>
<td>ARRMIE</td>
<td>Write 1 to ARRMCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>External trigger event</td>
<td>EXTTRIG</td>
<td>EXTTRIGIE</td>
<td>Write 1 to EXTTRIGCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Auto-reload register update OK</td>
<td>ARROK</td>
<td>ARROKIE</td>
<td>Write 1 to ARROKCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Capture/compare register update OK</td>
<td>CMPxOK</td>
<td>CMPxOKIE</td>
<td>Write 1 to CMPxOKCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Direction change to up(3)</td>
<td>UP</td>
<td>UPIE</td>
<td>Write 1 to UPCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Direction change to down(3)</td>
<td>DOWN</td>
<td>DOWNIE</td>
<td>Write 1 to DOWNCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Update Event</td>
<td>UE</td>
<td>UEIE</td>
<td>Write 1 to UECF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Repetition register update OK</td>
<td>REPOK</td>
<td>REPOKIE</td>
<td>Write 1 to REPOKCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

1. Each LPTIM event can wake up the device from Stop mode only if the LPTIM instance supports the wake-up from Stop mode feature. Refer to Section 43.3: LPTIM implementation.

2. If LPTIM does not implement any channel this event does not exist. Refer to Section 43.3: LPTIM implementation.

3. If LPTIM does not support encoder mode feature, this event does not exist. Refer to Section 43.3: LPTIM implementation.

### 43.7 LPTIM registers

Refer to Section 1.2: List of abbreviations for registers on page 101 for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).
### 43.7.1 LPTIM4 interrupt and status register (LPTIM4_ISR)

Address offset: 0x000  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:25</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bit 24   | **DIEROK**: Interrupt enable register update OK  
  DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register. |
| Bit 23:9 | Reserved, must be kept at reset value. |
| Bit 8    | **REPOK**: Repetition register update OK  
  REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register. |
| Bit 7    | **UE**: LPTIM update event occurred  
  UE is set by hardware to inform application that an update event was generated. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. |
| Bit 6    | **DOWN**: Counter direction change up to down  
  In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.  
  **Note**: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3: LPTIM implementation. |
| Bit 5    | **UP**: Counter direction change down to up  
  In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.  
  **Note**: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3: LPTIM implementation. |
| Bit 4    | **ARROK**: Autoreload register update OK  
  ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register. |
| Bit 3    | **CMP1OK**: Compare register 1 update OK  
  CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.
Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register’s value reached the LPTIM_ARR register’s value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register’s value. The CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match
1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register’s value

### 43.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR)  
(x = 1 to 3, 5, 6)

This description of the register can only be used for output compare mode. See next section for input capture mode.

Address offset: 0x000
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OK Res</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>REE OK</td>
<td>UE DOWN</td>
<td>UP ARRM</td>
<td>CMP1 OK</td>
<td>EXT TRIG</td>
<td>Arrm</td>
<td>CC1F</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK**: Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OK**: Compare register 2 update OK

CMP2OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR2 register has been successfully completed. CMP2OK flag can be cleared by writing 1 to the CMP2OKCF bit in the LPTIM_ICR register.

**Note:** If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bits 18:12 Reserved, must be kept at reset value.

Bits 9:0 Reserved, must be kept at reset value.
Bit 11  Reserved, must be kept at reset value.

Bit 10  Reserved, must be kept at reset value.

Bit 9  **CC2IF**: Compare 2 interrupt flag

  **If channel CC2 is configured as output:**
  
  The CC2IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

  0: No match
  
  1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR2 register's value

  *Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.*

Bit 8  **REPOK**: Repetition register update OK

  REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7  **UE**: LPTIM update event occurred

  UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6  **DOWN**: Counter direction change up to down

  In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

  *Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.*

Bit 5  **UP**: Counter direction change down to up

  In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

  *Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.*

Bit 4  **ARROK**: Autoreload register update OK

  ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3  **CMP1OK**: Compare register 1 update OK

  CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.
Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register’s value reached the LPTIM_ARR register’s value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

If channel CC1 is configured as output:

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register’s value. CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match
1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register’s value

### 43.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) 
(x = 1 to 3, 5, 6)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x000

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>DIEROK</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Res</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Arr</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Ext TRIG</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>CC1IF</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK**: Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OF**: Capture 2 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC2OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.
1: The counter value has been captured in LPTIM_CCR2 register while CC2IF flag was already set.

*Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.*
Bit 12 **CC1OF**: Capture 1 over-capture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC1OCF bit in the LPTIM_ICR register.
0: No over-capture has been detected.
1: The counter value has been captured in LPTIM_CCR1 register while CC1IF flag was already set.
*Note*: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 43.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Capture 2 interrupt flag
If channel CC2 is configured as input:
CC2IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR2 register. The corresponding interrupt or DMA request is generated if enabled. The CC2OF flag is set if the CC2IF flag was already high.
0: No input capture occurred
1: The counter value has been captured in the LPTIM_CCR2 register. (An edge has been detected on IC2 which matches the selected polarity). The CC2IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). The CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.
*Note*: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bit 8 **REPOK**: Repetition register update OK
REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred
UE is set by hardware to inform application that an update event was generated. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register.

Bit 6 **DOWN**: Counter direction change up to down
In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.
*Note*: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 5 **UP**: Counter direction change down to up
In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.
*Note*: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 4 **ARROK**: Autoreload register update OK
ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 Reserved, must be kept at reset value.
Bit 2 **EXTTRIG**: External trigger edge event
EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match
ARRM is set by hardware to inform application that LPTIM_CNT register’s value reached the LPTIM_ARR register’s value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: capture 1 interrupt flag
If channel CC1 is configured as input:
CC1IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR1 register. The corresponding interrupt or DMA request is generated if enabled. The CC1OF flag is set if the CC1IF flag was already high.
0: No input capture occurred
1: The counter value has been captured in the LPTIM_CCR1 register. (An edge has been detected on IC1 which matches the selected polarity). The CC1IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

### 43.7.4 LPTIM4 interrupt clear register (LPTIM4_ICR)

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF**: Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.

Note: *If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.*

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.

Note: *If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.*
Bit 4 ARROKCF: Autoreload register update OK clear flag
  Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register.

Bit 3 CMP1OKCF: Compare register 1 update OK clear flag
  Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 EXTTRIGCF: External trigger valid edge clear flag
  Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register.

Bit 1 ARRMCFCF: Autoreload match clear flag
  Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register.

Bit 0 CC1CF: Capture/compare 1 clear flag
  Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

43.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR)
(x = 1 to 3, 5, 6)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>DIEROKCF</td>
<td>CMP2OKCF</td>
<td>CMP1OKCF</td>
<td>EXTTRIGCF</td>
<td>ARRMCFCF</td>
<td>CC1CF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 DIEROKCF: Interrupt enable register update OK clear flag
  Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 CMP2OKCF: Compare register 2 update OK clear flag
  Writing 1 to this bit clears the CMP2OK flag in the LPTIM_ISR register.
  Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 CC2CF: Capture/compare 2 clear flag
  Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
  Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.
**Low-power timer (LPTIM)**

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clears the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clears the DOWN flag in the LPTIM_ISR register.
*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.*

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clears the UP flag in the LPTIM_ISR register.
*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.*

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register.

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register.

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register.

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

### 43.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR)

(x = 1 to 3, 5, 6)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

**Address offset**: 0x004

**Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>DIEROKCF: Interrupt enable register update OK clear flag</td>
</tr>
<tr>
<td>23</td>
<td>Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>13</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>12</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>9</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>6</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>4</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>3</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>2</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>1</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

**Bits 31:25**  Reserved, must be kept at reset value.

**Bit 24** **DIEROKCF**: Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

**Bits 23:16**  Reserved, must be kept at reset value.

**Bit 15**  Reserved, must be kept at reset value.

**Bit 14**  Reserved, must be kept at reset value.
### LPTIM4 interrupt enable register (LPTIM4_DIER)

Address offset: 0x008

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Bit Definitions

**Bit 13**  **CC2OCF**: Capture/compare 2 over-capture clear flag
- Writing 1 to this bit clears the CC2OF flag in the LPTIM_ISR register.
  - Note: If the LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

**Bit 12**  **CC1OCF**: Capture/compare 1 over-capture clear flag
- Writing 1 to this bit clears the CC1OF flag in the LPTIM_ISR register.
  - Note: If the LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 43.3.

**Bit 11**  Reserved, must be kept at reset value.

**Bit 10**  Reserved, must be kept at reset value.

**Bit 9**  **CC2CF**: Capture/compare 2 clear flag
- Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
  - Note: If the LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

**Bit 8**  **REPOKCF**: Repetition register update OK clear flag
- Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

**Bit 7**  **UECF**: Update event clear flag
- Writing 1 to this bit clears the UE flag in the LPTIM_ISR register.

**Bit 6**  **DOWNCF**: Direction change to down clear flag
- Writing 1 to this bit clears the DOWN flag in the LPTIM_ISR register.
  - Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.

**Bit 5**  **UPCF**: Direction change to UP clear flag
- Writing 1 to this bit clears the UP flag in the LPTIM_ISR register.
  - Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.

**Bit 4**  **ARROKCF**: Autoreload register update OK clear flag
- Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register.

**Bit 3**  Reserved, must be kept at reset value.

**Bit 2**  **EXTTRIGCF**: External trigger valid edge clear flag
- Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register.

**Bit 1**  **ARRMCF**: Autoreload match clear flag
- Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register.

**Bit 0**  **CC1CF**: Capture/compare 1 clear flag
- Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.
  - Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.
Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.
### 43.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER)  
(x = 1 to 3, 5, 6)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x008  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

- **Bit 23** UEDE: Update event DMA request enable  
  0: UE DMA request disabled. Writing ‘0’ to the UEDE bit resets the associated ue_dma_req signal.  
  1: UE DMA request enabled  
  *Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 43.3.*

- **Bit 22** Reserved, must be kept at reset value.

- **Bit 21** Reserved, must be kept at reset value.

- **Bit 20** Reserved, must be kept at reset value.

- **Bit 19** CMP2OKIE: Compare register 2 update OK interrupt enable  
  0: CMPOK register 2 interrupt disabled  
  1: CMPOK register 2 interrupt enabled  
  *Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.*

Bits 18:12 Reserved, must be kept at reset value.

- **Bit 11** Reserved, must be kept at reset value.

- **Bit 10** Reserved, must be kept at reset value.

- **Bit 9** CC2IE: Capture/compare 2 interrupt enable  
  0: Capture/compare 2 interrupt disabled  
  1: Capture/compare 2 interrupt enabled  
  *Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.*

- **Bit 8** REPOKIE: Repetition register update OK interrupt Enable  
  0: Repetition register update OK interrupt disabled  
  1: Repetition register update OK interrupt enabled

- **Bit 7** UEIE: Update event interrupt enable  
  0: Update event interrupt disabled  
  1: Update event interrupt enabled
Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
   0: DOWN interrupt disabled
   1: DOWN interrupt enabled

   *Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.*

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
   0: UP interrupt disabled
   1: UP interrupt enabled

   *Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.*

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
   0: ARROK interrupt disabled
   1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
   0: CMPOK register 1 interrupt disabled
   1: CMPOK register 1 interrupt enabled

Bit 2 **EXTRTRIGIE**: External trigger valid edge Interrupt Enable
   0: EXTRTRIG interrupt disabled
   1: EXTRTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
   0: ARRM interrupt disabled
   1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
   0: Capture/compare 1 interrupt disabled
   1: Capture/compare 1 interrupt enabled

**Caution:** The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

### 43.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3, 5, 6)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:28  Reserved, must be kept at reset value.
Bit 27  Reserved, must be kept at reset value.
Bit 26  Reserved, must be kept at reset value.
Bit 25  **CC2DE**: Capture/compare 2 DMA request enable
- 0: CC2 DMA request disabled. Writing '0' to the CC2DE bit resets the associated ic2 dma_req signal.
- 1: CC2 DMA request enabled
  
  **Note:** *If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.*

Bit 24  Reserved, must be kept at reset value.
Bit 23  **UEDE**: Update event DMA request enable
- 0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
- 1: UE DMA request enabled
  
  **Note:** *If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 43.3.*

Bits 22:17  Reserved, must be kept at reset value.
Bit 16  **CC1DE**: Capture/compare 1 DMA request enable
- 0: CC1 DMA request disabled. Writing '0' to the CC1DE bit resets the associated ic1_dma_req signal.
- 1: CC1 DMA request enabled
  
  **Note:** *If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 43.3.*

Bit 15  Reserved, must be kept at reset value.
Bit 14  Reserved, must be kept at reset value.
Bit 13  **CC2OIE**: Capture/compare 2 over-capture interrupt enable
- 0: CC2 over-capture interrupt disabled
- 1: CC2 over-capture interrupt enabled
  
  **Note:** *If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.*

Bit 12  **CC1OIE**: Capture/compare 1 over-capture interrupt enable
- 0: CC1 over-capture interrupt disabled
- 1: CC1 over-capture interrupt enabled
  
  **Note:** *If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 43.3.*

Bit 11  Reserved, must be kept at reset value.
Bit 10  Reserved, must be kept at reset value.
Bit 9  **CC2IE**: Capture/compare 2 interrupt enable
- 0: Capture/compare 2 interrupt disabled
- 1: Capture/compare 2 interrupt enabled
  
  **Note:** *If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.*

Bit 8  **REPOKIE**: Repetition register update OK interrupt Enable
- 0: Repetition register update OK interrupt disabled
- 1: Repetition register update OK interrupt enabled

Bit 7  **UEIE**: Update event interrupt enable
- 0: Update event interrupt disabled
- 1: Update event interrupt enabled
Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled

*Note*: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled

*Note*: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

**Caution**: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

### 43.7.10 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field</th>
<th>Width</th>
<th>Read/Write</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>ENC</td>
<td>1</td>
<td>rw</td>
<td>Counts the number of up events</td>
</tr>
<tr>
<td>30</td>
<td>COUNT MODE</td>
<td>1</td>
<td>rw</td>
<td>Selects count mode</td>
</tr>
<tr>
<td>29</td>
<td>PRE LOAD</td>
<td>1</td>
<td>rw</td>
<td>Preloads the value</td>
</tr>
<tr>
<td>28</td>
<td>WAV POL</td>
<td>1</td>
<td>rw</td>
<td>Selects wave polynomial</td>
</tr>
<tr>
<td>27</td>
<td>WAV</td>
<td>1</td>
<td>rw</td>
<td>Sets the wave</td>
</tr>
<tr>
<td>26</td>
<td>TIMOUT</td>
<td>1</td>
<td>rw</td>
<td>Sets the timeout</td>
</tr>
<tr>
<td>25</td>
<td>TRIGEN[1:0]</td>
<td>2</td>
<td>rw</td>
<td>Selects trigger event</td>
</tr>
<tr>
<td>24</td>
<td>TRIGSEL[2:0]</td>
<td>3</td>
<td>rw</td>
<td>Selects trigger event</td>
</tr>
<tr>
<td>23</td>
<td>PRESC[2:0]</td>
<td>2</td>
<td>rw</td>
<td>Selects prescaler</td>
</tr>
<tr>
<td>22</td>
<td>TRGFLT[1:0]</td>
<td>2</td>
<td>rw</td>
<td>Selects trigger filter</td>
</tr>
<tr>
<td>21</td>
<td>CKFLT[1:0]</td>
<td>2</td>
<td>rw</td>
<td>Selects clock filter</td>
</tr>
<tr>
<td>20</td>
<td>CKPOL[1:0]</td>
<td>2</td>
<td>rw</td>
<td>Selects clock polarity</td>
</tr>
<tr>
<td>19</td>
<td>CKSEL</td>
<td>1</td>
<td>rw</td>
<td>Selects clock</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.
Bit 24 **ENC**: Encoder mode enable
The ENC bit controls the Encoder mode
0: Encoder mode disabled
1: Encoder mode enabled

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3.*

Bit 23 **COUNTMODE**: counter mode enabled
The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:
0: the counter is incremented following each internal clock pulse
1: the counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode
The PRELOAD bit controls the LPTIM_ARR, LPTIM_RCR and the LPTIM_CCRx registers update modality
0: Registers are updated after each APB bus write access
1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity
The WAVPOL bit controls the output polarity
0: The LPTIM output reflects the compare results between LPTIM_CNT and LPTIM_CCRx registers
1: The LPTIM output reflects the inverse of the compare results between LPTIM_CNT and LPTIM_CCRx registers

*Note: If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to Section 43.3.*

Bit 20 **WAVE**: Waveform shape
The WAVE bit controls the output shape
0: Deactivate Set-once mode
1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable
The TIMOUT bit controls the Timeout feature
0: A trigger event arriving when the timer is already started is ignored
1: A trigger event arriving when the timer is already started resets and restarts the LPTIM counter and the repetition counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity
The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:
00: software trigger (counting start is initiated by software)
01: rising edge is the active edge
10: falling edge is the active edge
11: both edges are active edges

Bit 16 Reserved, must be kept at reset value.
Bits 15:13 **TRIGSEL[2:0]:** Trigger selector
The TRIGSEL bits select the trigger source that serves as a trigger event for the LPTIM among the below 8 available sources:
000: lptim_ext_trig0
001: lptim_ext_trig1
010: lptim_ext_trig2
011: lptim_ext_trig3
100: lptim_ext_trig4
101: lptim_ext_trig5
110: lptim_ext_trig6
111: lptim_ext_trig7
See [Section 43.4.3: LPTIM input and trigger mapping](#) for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler
The PRESC bits configure the prescaler division factor. It can be one among the following division factors:
000: /1
001: /2
010: /4
011: /8
100: /16
101: /32
110: /64
111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger
The TRGFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature
00: any trigger active level change is considered as a valid trigger
01: trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.
10: trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.
11: trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock
The CKFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature
00: any external clock signal level change is considered as a valid transition
01: external clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
10: external clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
11: external clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.
43.7.11 LPTIM control register (LPTIM_CR)

Address offset: 0x010
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 RSTARE: Reset after read enable
This bit is set and cleared by software. When RSTARE is set to ‘1’, any read access to LPTIM_CNT register asynchronously resets LPTIM_CNT register content.
This bit can be set only when the LPTIM is enabled.

Bit 3 COUNTRST: Counter reset
This bit is set by software and cleared by hardware. When set to ‘1’ this bit triggers a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock may be different from APB clock).
This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Caution: COUNTRST must never be set to ‘1’ by software before it is already cleared to ‘0’ by hardware. Software must consequently check that COUNTRST bit is already cleared to ‘0’ before attempting to set it to ‘1’.
Bit 2  CNTSTRT: Timer start in Continuous mode  
This bit is set by software and cleared by hardware.  
In case of software start (TRIGEN[1:0] = ‘00’), setting this bit starts the LPTIM in Continuous mode.  
If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.  
If this bit is set when a single pulse mode counting is ongoing, then the timer does not stop at the next match between the LPTIM.ARR and LPTIM.CNT registers and the LPTIM counter keeps counting in Continuous mode.  
This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Bit 1  SNGSTRT: LPTIM start in Single mode  
This bit is set by software and cleared by hardware.  
In case of software start (TRIGEN[1:0] = ‘00’), setting this bit starts the LPTIM in single pulse mode.  
If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.  
If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM stops at the following match between LPTIM.ARR and LPTIM.CNT registers.  
This bit can only be set when the LPTIM is enabled. It is automatically reset by hardware.

Bit 0  ENABLE: LPTIM enable  
The ENABLE bit is set and cleared by software.  
0: LPTIM is disabled. Writing ‘0’ to the ENABLE bit resets all the DMA request signals (input capture and update event DMA requests).  
1: LPTIM is enabled

43.7.12  LPTIM compare register 1 (LPTIM_CCR1)  
Address offset: 0x014  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  CCR1[15:0]: Capture/compare 1 value  
If channel CC1 is configured as output:  
CCR1 is the value to be loaded in the capture/compare 1 register.  
Depending on the PRELOAD option, the CCR1 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOD bit is reset.  
The capture/compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on OC1 output.  
If channel CC1 is configured as input:  
CCR1 becomes read-only, it contains the counter value transferred by the last input capture 1 event.  
The LPTIM_CCR1 register is read-only and cannot be programmed.  
If LPTIM does not implement any channel:  
The compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on LPTIM output.
Caution: The LPTIM_CCR1 register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

43.7.13 LPTIM autoreload register (LPTIM.ARR)

Address offset: 0x018
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ARR[15:0]: Auto reload value

Bits 15:0

Bits 31:16 Reserved, must be kept at reset value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

43.7.14 LPTIM counter register (LPTIM.CNT)

Address offset: 0x01C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CNT[15:0]: Counter value

Bits 15:0

Bits 31:16 Reserved, must be kept at reset value.

Caution: When the LPTIM is running with an asynchronous clock, reading the LPTIM_CNT register may return unreliable values. So in this case it is necessary to perform two consecutive read accesses and verify that the two returned values are identical.
### 43.7.15 LPTIM configuration register 2 (LPTIM\_CFGR2)

Address offset: 0x024  
Reset value: 0x0000 0000

| Bit 31:22 | Address offset: 0x024  
Reset value: 0x0000 0000 |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bin 21:0</td>
<td>IC2SEL[1:0]: LPTIM input capture 2 selection</td>
</tr>
<tr>
<td>Bin 19:18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bin 17:16</td>
<td>IC1SEL[1:0]: LPTIM input capture 1 selection</td>
</tr>
<tr>
<td>Bin 15:6</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bin 5:4</td>
<td>IN2SEL[1:0]: LPTIM input 2 selection</td>
</tr>
<tr>
<td>Bin 3:2</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bin 1:0</td>
<td>IN1SEL[1:0]: LPTIM input 1 selection</td>
</tr>
</tbody>
</table>

- **IC2SEL[1:0]:** Control the LPTIM Input capture 2 multiplexer, which connects LPTIM Input capture 2 to one of the available inputs.  
  - 00: lptim\_ic2\_mux0  
  - 01: lptim\_ic2\_mux1  
  - 10: lptim\_ic2\_mux2  
  - 11: lptim\_ic2\_mux3  

For connection details refer to Section 43.4.3: LPTIM input and trigger mapping.

- **IC1SEL[1:0]:** Control the LPTIM Input capture 1 multiplexer, which connects LPTIM Input capture 1 to one of the available inputs.  
  - 00: lptim\_ic1\_mux0  
  - 01: lptim\_ic1\_mux1  
  - 10: lptim\_ic1\_mux2  
  - 11: lptim\_ic1\_mux3  

For connection details refer to Section 43.4.3: LPTIM input and trigger mapping.

- **IN2SEL[1:0]:** Control the LPTIM input 2 multiplexer, which connects LPTIM input 2 to one of the available inputs.  
  - 00: lptim\_in2\_mux0  
  - 01: lptim\_in2\_mux1  
  - 10: lptim\_in2\_mux2  
  - 11: lptim\_in2\_mux3  

For connection details refer to Section 43.4.3: LPTIM input and trigger mapping.

- **IN1SEL[1:0]:** Control the LPTIM input 1 multiplexer, which connects LPTIM input 1 to one of the available inputs.  
  - 00: lptim\_in1\_mux0  
  - 01: lptim\_in1\_mux1  
  - 10: lptim\_in1\_mux2  
  - 11: lptim\_in1\_mux3  

For connection details refer to Section 43.4.3: LPTIM input and trigger mapping.
### 43.7.16 LPTIM repetition register (LPTIM_RCR)

Address offset: 0x028  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>REP[7:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.  
Bits 7:0 **REP[7:0]**: Repetition register value  
REP is the repetition value for the LPTIM.

**Caution:** The LPTIM_RCR register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’). When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the auto reload match event, otherwise an unpredictable behavior may occur.

### 43.7.17 LPTIM capture/compare mode register 1 (LPTIM_CCMR1)

Address offset: 0x02C  
Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (PWM mode). The direction of a channel is defined by configuring the corresponding CCxSEL bits.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>CC2P[1:0]</td>
<td>CC2E</td>
<td>CC2 SEL</td>
<td>CC1P[1:0]</td>
<td>CC1E</td>
<td>CC1 SEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:30  Reserved, must be kept at reset value.

Bits 29:28  **IC2F[1:0]: Input capture 2 filter**
This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

- 00: any external input capture signal level change is considered as a valid transition
- 01: external input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
- 10: external input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
- 11: external input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 27:26  Reserved, must be kept at reset value.

Bits 25:24  **IC2PSC[1:0]: Input capture 2 prescaler**
This bitfield defines the ratio of the prescaler acting on the CC2 input (IC2).

- 00: no prescaler, capture is done each time an edge is detected on the capture input
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 23:20  Reserved, must be kept at reset value.

Bits 19:18  **CC2P[1:0]: Capture/compare 2 output polarity.**
Condition: CC2 as output
Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.
- 0: OC2 active high
- 1: OC2 active low

Condition: CC2 as input
This field is used to select the IC2 polarity for capture operations.
- 00: rising edge, circuit is sensitive to IC2 rising edge
- 01: falling edge, circuit is sensitive to IC2 falling edge
- 10: reserved, do not use this configuration.
- 11: both edges, circuit is sensitive to both IC2 rising and falling edges.

Bit 17  **CC2E**: Capture/compare 2 output enable.
Condition: CC2 as output
- 0: Off - OC2 is not active. Writing '0' to the CC2E bit resets the uc_dma_req signal only if all the other LPTIM channels are disabled.
- 1: On - OC2 signal is output on the corresponding output pin

Condition: CC2 as input
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 2 (LPTIM_CCR2) or not.
- 0: Capture disabled. Writing '0' to the CC2E bit resets the associated ic2_dma_req signal.
- 1: Capture enabled.

Bit 16  **CC2SEL**: Capture/compare 2 selection
This bitfield defines the direction of the channel, input (capture) or output mode.
- 0: CC2 channel is configured in output PWM mode
- 1: CC2 channel is configured in input capture mode

Bits 15:14  Reserved, must be kept at reset value.
Bits 13:12 IC1F[1:0]: Input capture 1 filter
This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.
00: any external input capture signal level change is considered as a valid transition
01: external input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
10: external input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
11: external input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 IC1PSC[1:0]: Input capture 1 prescaler
This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 CC1P[1:0]: Capture/compare 1 output polarity.
Condition: CC1 as output
Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.
0: OC1 active high, the LPTIM output reflects the compare results between LPTIM.ARR and LPTIM.CCRx registers
1: OC1 active low, the LPTIM output reflects the inverse of the compare results between LPTIM.ARR and LPTIM.CCRx registers

Condition: CC1 as input
This field is used to select the IC1 polarity for capture operations.
00: rising edge, circuit is sensitive to IC1 rising edge
01: falling edge, circuit is sensitive to IC1 falling edge
10: reserved, do not use this configuration.
11: both edges, circuit is sensitive to both IC1 rising and falling edges.

Bit 1 CC1E: Capture/compare 1 output enable.
Condition: CC1 as output
0: Off - OC1 is not active. Writing '0' to the CC1E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.
1: On - OC1 signal is output on the corresponding output pin

Condition: CC1 as input
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (LPTIM.CCR1) or not.
0: Capture disabled. Writing '0' to the CC1E bit resets the associated ic1_dma_req signal.
1: Capture enabled.

Bit 0 CC1SEL: Capture/compare 1 selection
This bitfield defines the direction of the channel input (capture) or output mode.
0: CC1 channel is configured in output PWM mode
1: CC1 channel is configured in input capture mode

Caution: After a write to the LPTIM.CCMRx register, a new write operation to the same register can only be performed after a delay that must be equal or greater than the value of (PRESC × 3)
kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4,..128). Any successive write violating this delay, leads to unpredictable results.

Caution: The CCxSEL, ICxF[1:0], CCxP[1:0] and ICxPSC[1:0] fields must only be modified when the channel x is disabled (CCxE bit reset to 0).

If LPTIM does not implement any channel this register is reserved. Refer to Section 43.3.

43.7.18 LPTIM compare register 2 (LPTIM_CCR2)

Address offset: 0x034
Reset value: 0x0000 0000

Caution: The LPTIM_CCR2 register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

Note: If the LPTIM implements less than 2 channels this register is reserved. Refer to Section 43.3: LPTIM implementation.

43.7.19 LPTIM register map

The following table summarizes the LPTIM registers.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>CCR2[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>LPTIM4_ISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 466. LPTIM register map and reset values
### Table 466. LPTIM register map and reset values (continued)

| Offset  | Register name                               | Bit 31 | Bit 30 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---------|---------------------------------------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0x000   | LPTIMx_ISR (x = 1 to 3, 5, 6) Output compare mode | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
|         | Reset value                                 | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| 0x004   | LPTIM4_ICR                                  | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
|         | Reset value                                 | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| 0x004   | LPTIM4_ICR                                  | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
|         | Reset value                                 | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| 0x008   | LPTIM4_DIER                                 | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
|         | Reset value                                 | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| 0x008   | LPTIM4_DIER                                 | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
|         | Reset value                                 | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| 0x00C   | LPTIM_CFGR                                  | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
|         | Reset value                                 | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| 0x010   | LPTIM_CR                                   | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
|         | Reset value                                 | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
| 0x014   | LPTIM_CCR1                                  | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
|         | Reset value                                 | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |

<table>
<thead>
<tr>
<th>LPTIMx_ISR (x = 1 to 3, 5, 6) Output compare mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>0x000</td>
</tr>
<tr>
<td>0x004</td>
</tr>
<tr>
<td>0x008</td>
</tr>
<tr>
<td>0x00C</td>
</tr>
<tr>
<td>0x010</td>
</tr>
<tr>
<td>0x014</td>
</tr>
</tbody>
</table>
Table 466. LPTIM register map and reset values (continued)

| Offset | Register name       | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|
| 0x018  | LPTIM_ARR           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
|        | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |   |
| 0x01C  | LPTIM_CNT           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
|        | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |   |
| 0x024  | LPTIM_CFGR2         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
|        | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |   |
| 0x028  | LPTIM_RCR           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
|        | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |   |
| 0x02C  | LPTIM_CCMR1(4)      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
|        | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |   |
| 0x034  | LPTIM_CCR2(5)       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
|        | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |   |

1. If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3: LPTIM implementation.
2. If LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 43.3: LPTIM implementation.
3. If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to Section 43.3: LPTIM implementation.
4. If LPTIM does not implement any channel this register is reserved. Refer to Section 43.3: LPTIM implementation.
5. If the LPTIM implements less than 2 channels this register is reserved. Refer to Section 43.3: LPTIM implementation.

Refer to Section 2.3 on page 111 for the register boundary addresses.
44 Independent watchdog (IWDG)

44.1 Introduction

The independent watchdog (IWDG) peripheral offers a high safety level, thanks to its capability to detect malfunctions due to software or hardware failures.

The IWDG is clocked by an independent clock, and stays active even if the main clock fails.

In addition, the watchdog function is performed in the $V_{DD}$ voltage domain, allowing the IWDG to remain functional even in low-power modes. Refer to Section 44.3 to check the capability of the IWDG in this product.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, making it very reliable to detect any unexpected behavior.

44.2 IWDG main features

- 12-bit down-counter
- Dual voltage domain, thus enabling operation in low-power modes
- Independent clock
- Early wake-up interrupt generation
- Reset generation
  - In case of timeout
  - In case of refresh outside the expected window

44.3 IWDG implementation

<table>
<thead>
<tr>
<th>IWDG modes/features</th>
<th>IWDG</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSI used as IWDG kernel clock (iwdg_ker_ck)</td>
<td>X</td>
</tr>
<tr>
<td>Window function</td>
<td>X</td>
</tr>
<tr>
<td>Early wake-up interrupt generation</td>
<td>X</td>
</tr>
<tr>
<td>System reset generation $^{(2)}$</td>
<td>X</td>
</tr>
<tr>
<td>Capability to work in system Stop</td>
<td>X</td>
</tr>
<tr>
<td>Capability to work in system Standby</td>
<td>X</td>
</tr>
<tr>
<td>Capability to generate an interrupt in system Stop</td>
<td>X</td>
</tr>
<tr>
<td>Capability to generate an interrupt in system Standby</td>
<td>-</td>
</tr>
<tr>
<td>Capability to be frozen when the microcontroller enters in Debug mode $^{(3)}$</td>
<td>X</td>
</tr>
<tr>
<td>Option bytes to control the activity in Stop mode $^{(4)}$</td>
<td>X</td>
</tr>
<tr>
<td>Option bytes to control the activity in Standby mode $^{(5)}$</td>
<td>X</td>
</tr>
<tr>
<td>Option bytes to control the Hardware mode $^{(6)}$</td>
<td>X</td>
</tr>
</tbody>
</table>

1. ‘X’ = supported, ‘-’ = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via DBG_IWDG_STOP in DBG section.
4. Controlled via the option byte IWDG_STOP in FLASH section.
5. Controlled via the option byte IWDG_STDBY in FLASH section.
6. Controlled via the option byte IWDG_SW in FLASH section.

## 44.4 IWDG functional description

### 44.4.1 IWDG block diagram

*Figure 609* shows the functional blocks of the independent watchdog module.

*Figure 609. Independent watchdog block diagram*

The register and IRQ interfaces are located into the V\textsubscript{CORE} voltage domain. The watchdog function itself is located into the V\textsubscript{DD} voltage domain to remain functional in low-power modes. See *Section 44.3* for IWDG capabilities.

The register and IRQ interfaces are mainly clocked by the APB clock (iwdg\_pclk), while the watchdog function is clocked by a dedicated kernel clock (iwdg\_ker\_ck). A synchronization mechanism makes the data exchange between the two domains possible. Note that most of the registers located in the register interface are shadowed into the V\textsubscript{DD} voltage domain.

The IWDG down-counter (IWDCNT) is clocked by the prescaled clock (presc\_ck). The prescaled clock is generated from the kernel clock iwdg\_ker\_ck divided by the prescaler, according to PR[3:0] bitfield.
### 44.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in Table 468.

#### Table 468. IWDG internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iwgd_ker_ck</td>
<td>Input</td>
<td>IWDG kernel clock</td>
</tr>
<tr>
<td>iwgd_ker_req</td>
<td>Input</td>
<td>IWDG kernel clock request</td>
</tr>
<tr>
<td>iwgd_pclk</td>
<td>Input</td>
<td>IWDG APB clock</td>
</tr>
<tr>
<td>iwgd_out_rst</td>
<td>Output</td>
<td>IWDG reset output</td>
</tr>
<tr>
<td>iwgd_in_rst</td>
<td>Input</td>
<td>IWDG reset input</td>
</tr>
<tr>
<td>iwgd_wkup</td>
<td>Output</td>
<td>IWDG wake-up event</td>
</tr>
<tr>
<td>iwgd_it</td>
<td>Output</td>
<td>IWDG early wake-up interrupt</td>
</tr>
</tbody>
</table>

### 44.4.3 Software and hardware watchdog modes

The watchdog modes allow the application to select the way the IWDG is enabled, either by software commands (Software watchdog mode), or automatically (Hardware watchdog mode). All other functions work similarly for both Software and Hardware modes.

The Software watchdog mode is the default working mode. The independent watchdog is started by writing the value 0x0000 CCCC into the **IWDG key register (IWDG_KR)**, and the IWDCNT starts counting down from the reset value (0xFFF).

In the Hardware watchdog mode the independent watchdog is started automatically at power-on, or every time it is reset (via iwgd_in_rst). The IWDCNT down-counter starts counting down from the reset value 0xFFF. The hardware watchdog mode feature is enabled through the device option bits, see Section 44.3 for details.

When the IWDG is enabled the ONF flag is set to 1.

When the IWDCNT reaches 0x000, a reset signal is generated (iwgd_out_rst asserted).

Whenever the key value 0x0000 AAAA is written in the **IWDG key register (IWDG_KR)**, the IWGD_RLR value is reloaded into the IWDCNT, and the watchdog reset is prevented.

Due to re-synchronization delays, the IWDG must be refreshed before the IWDCNT down-counter reaches 1.

Once started, the IWDG can be stopped only when it is reset (iwgd_in_rst asserted).

As shown in Figure 610, when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].
1. If window option activated.

If the IWDG is not refreshed before the IWDCNT reaches 1, then the IWDG generates a reset (i.e. iwdg_out_rst asserted). In return, the RCC resets the IWDG (assertion of iwdg_in_rst) to clear the reset source.

44.4.4 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the IWDG window register (IWDG_WINR).

If the reload operation is performed while the counter is greater than WIN[11:0] + 1 a reset is generated. WIN[11:0] is located in the IWDG window register (IWDG_WINR). As shown in Figure 611, the reset is generated one period of presc_ck after the unexpected refresh command.

The default value of the IWDG window register (IWDG_WINR) is 0x0000 0xFFF, so, if not updated, the window option is disabled.

As soon as the window value changes, the down-counter (IWDCNT) is reloaded with the RL[11:0] value to ease the estimation for where the next refresh must take place.
Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the IWDG key register (IWDG_KR).
2. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
3. Write the IWDG prescaler by programming IWDG prescaler register (IWDG_PR).
4. Write the IWDG reload register (IWDG_RLR).
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the IWDG early wake-up interrupt register (IWDG_EWCR).
6. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with the RL[11:0] value.
7. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
8. Write 0x0000 0000 into IWDG key register (IWDG_KR) to write-protect registers.

Note: Step 7 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:
1. Enable the IWDG by writing 0x0000 CCCC in the IWDG key register (IWDG_KR).
2. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
3. Write the prescaler by programming the IWDG prescaler register (IWDG_PR).
4. Write the IWDG reload register (IWDG_RLR).
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the IWDG early wake-up interrupt register (IWDG_EWCR).
6. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
7. Refresh the counter with RL[11:0] value, and write-protect registers by writing 0x0000 AAAA into IWDG key register (IWDG_KR).

**Updating the window comparator**

It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:
1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with RL[11:0] value.
3. Wait for WVU = 0
4. Lock registers by writing IWDG_KR to 0x0000 0000

Step 3 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

*Figure 612* shows this sequence. As soon as the IWDG_WINR register is written, the WVU flag goes high. The new window value and the reload of IWDCNT with RL[11:0] are effective on the next rising edge of presc_ck. The WVU flag goes back to 0, in the worst case, two kernel clock periods later. So WVU remains high at most one period of presc_ck, plus two periods of the kernel clock.

*Figure 612. Example of window comparator update*

---

(1) Write 0x0000 5555 into IWDG_KR to unlock registers
(2) Write new value into IWDG_WINR
(3) Write 0x0000 0000 into IWDG_KR to lock registers without refresh
44.4.5 Debug

When the processor enters into Debug mode (core halted), the IWDCNT down-counter either continues to work normally or stops, depending on debug capability of the product. Refer to Section 44.3 for details on the capabilities of this product.

44.4.6 Register access protection

Write accesses to IWDG prescaler register (IWDG_PR), IWDG reload register (IWDG_RLR), IWDG early wake-up interrupt register (IWDG_EWCR) and IWDG window register (IWDG_WINR) are protected. To modify them, first write 0x0000 5555 in the IWDG key register (IWDG_KR). A write access to this register with a different value breaks the sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is ongoing.

44.5 IWDG low-power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low-power modes. Refer to Section 44.3 for details.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. IWDG interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The IWDG remains active or not, depending on option bytes configuration. Refer to Section 44.3 for details. IWDG interrupts cause the device exit the Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The IWDG remains active or not, depending on option bytes configuration. Refer to Section 44.3 for details. IWDG interrupts do not make the device to exit from Standby mode.</td>
</tr>
</tbody>
</table>

44.6 IWDG interrupts

The IWDG offers the possibility to generate an early interrupt depending on the value of the down-counter. The early interrupt is enabled by setting the EWIE bit of the IWDG early wake-up interrupt register (IWDG_EWCR) to 1.

A comparator value (EWIT[11:0]) allows the application to define at which position the early interrupt must be generated.

When the IWDCNT down-counter reaches the value of EWIT[11:0] - 1, the iwdg_wkup is activated, making it possible for the system to exit from low-power mode if needed.

When the APB clock is available, the iwdg_it is activated as well.

In addition the flag EWIF of the IWDG status register (IWDG_SR) is set to 1.

The EWI interrupt is acknowledged by writing ‘1’ to the EWIC bit in the IWDG early wake-up interrupt register (IWDG_EWCR).
Writing into the IWDG_EWCR register also triggers a refresh of the down-counter (IWDCNT) with the reload value RL[11:0].

**Figure 613. Independent watchdog interrupt timing diagram**

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the watchdog reset is generated.

**Changing the early wake-up comparator value**

It is possible to change the early wake-up comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Enable or disable the early wake-up interrupt, and/or program the early wake-up comparator, by writing the proper values into the IWDG early wake-up interrupt register (IWDG_EWCR).
3. Wait for EWU = ’0’, EWU is located into the IWDG status register (IWDG_SR).
4. Write-protect registers by writing 0x0000 0000 to IWDG key register (IWDG_KR).

Step 3 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

*Figure 613* shows this sequence. As soon as the IWDG_EWCR register is written, the EWU flag goes high. The new comparator value and the reload of IWDCNT with RL[11:0] are effective on the next rising edge of presc_ck. The EWU flag goes back to 0, in the worst case, two kernel clock periods later. So, EWU remains high at most one period of presc_ck, plus two periods of the kernel clock.
**44.7 IWDG registers**

Refer to *Section 1.2 on page 101* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

Most of the registers located into the register interface are shadowed into the VDD voltage domain. When the iwdg_in_rst is asserted, the watchdog logic and the shadow registers located into the VDD voltage domain are reset.

When the application reads back a watchdog register, the hardware transfers the value of the corresponding shadow register to the register interface.

When the application writes a watchdog register, the hardware updates the corresponding shadow register.

---

**Table 470. IWDG interrupt request**

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Interrupt clear method</th>
<th>Interrupt enable control bit</th>
<th>Activated interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>IWDCNT reaches EWIT value</td>
<td>EWIF</td>
<td>Writing EWIC to ‘1’</td>
<td>EWIE</td>
<td>iwdg_it</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>iwdg_wkup_it</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y(2)</td>
</tr>
</tbody>
</table>

1. Generated when a clock is present on iwdg_pclk input.
2. Generated when a clock is present on iwdg_ker_ck input.

---

**Table 470 summarizes the IWDG interrupt request.**

---

**Figure 614. Example of early wake-up comparator update**

- iwdg_ker_ck
- presc_ck
- IWDCNT
- EWIT[11:0]
- EWU
- CPU accesses
- EWIT[11:0]

(1) Write 0x0000 5555 into IWDG_KR to unlock registers
(2) Write new value into IWDG_EWCR
(3) Write 0x0000 0000 into IWDG_KR to lock registers without refresh

---

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Interrupt clear method</th>
<th>Interrupt enable control bit</th>
<th>Activated interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>IWDCNT reaches EWIT value</td>
<td>EWIF</td>
<td>Writing EWIC to ‘1’</td>
<td>EWIE</td>
<td>iwdg_it</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y(1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>iwdg_wkup_it</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y(2)</td>
</tr>
</tbody>
</table>
44.7.1 IWDG key register (IWDG_KR)

Address offset: 0x00
Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEY[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **KEY[15:0]**: Key value (write only, read 0x0000)

These bits can be used for several functions, depending upon the value written by the application:
- 0xAAAA: reloads the RL[11:0] value into the IWDCNT down-counter (watchdog refresh), and write-protects registers. This value must be written by software at regular intervals, otherwise the watchdog generates a reset when the counter reaches 0.
- 0x5555: enables write-accesses to the registers.
- 0xCCCC: enables the watchdog (except if the hardware watchdog option is selected) and write-protects registers.
- Values different from 0x5555: write-protects registers.

Note that only IWDG_PR, IWDG_RLR, IWDG_EWCR and IWDG_WINR registers have a write-protection mechanism.

44.7.2 IWDG prescaler register (IWDG_PR)

Address offset: 0x04
Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PR[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:4  Reserved, must be kept at reset value.
### Bits 3:0 PR[3:0]: Prescaler divider

These bits are write access protected, see Section 44.4.6. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the IWDG status register (IWDG_SR) must be reset to be able to change the prescaler divider.

<table>
<thead>
<tr>
<th>Bin</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>divider / 4</td>
</tr>
<tr>
<td>0001</td>
<td>divider / 8</td>
</tr>
<tr>
<td>0010</td>
<td>divider / 16</td>
</tr>
<tr>
<td>0011</td>
<td>divider / 32</td>
</tr>
<tr>
<td>0100</td>
<td>divider / 64</td>
</tr>
<tr>
<td>0101</td>
<td>divider / 128</td>
</tr>
<tr>
<td>0110</td>
<td>divider / 256</td>
</tr>
<tr>
<td>0111</td>
<td>divider / 512</td>
</tr>
<tr>
<td>Others</td>
<td>divider / 1024</td>
</tr>
</tbody>
</table>

*Note: Reading this register returns the prescaler value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG status register (IWDG_SR) is reset.*

#### 44.7.3 IWDG reload register (IWDG_RLR)

**Address offset:** 0x08  
**Reset value:** 0x0000 0FFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

**Bits 11:0 RL[11:0]: Watchdog counter reload value**

These bits are write access protected, see Section 44.4.6. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAA is written in the IWDG key register (IWDG_KR). The watchdog counter counts down from this value. The timeout period is a function of this value and the prescaler.clock. It is not recommended to set RL[11:0] to a value lower than 2.

The RVU bit in the IWDG status register (IWDG_SR) must be reset to be able to change the reload value.

*Note: Reading this register returns the reload value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.*

#### 44.7.4 IWDG status register (IWDG_SR)

**Address offset:** 0x0C  
**Reset value:** 0x0000 0000 (0xFFFF FEFF)

This register contains various status flags. Note that the mask value between parenthesis means that the reset value of ONF bit is not defined. When the IWDG is configured in
software mode, the reset value of ONF bit is 0, when the IWDG is configured in hardware
mode, the reset value of ONF bit is 1.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:15  Reserved, must be kept at reset value.

Bit 14  EWIF: Watchdog early interrupt flag
   This bit is set to ‘1’ by hardware in order to indicate that an early interrupt is pending. This bit
   must be cleared by the software by writing the bit EWIC of IWDG_EWCR register to ‘1’.

Bits 13:9  Reserved, must be kept at reset value.

Bit 8  ONF: Watchdog enable status bit
   Set to ‘1’ by hardware as soon as the IWDG is started. In software mode, it remains to ‘1’ until
   the IWDG is reset. In hardware mode, this bit is always set to ‘1’.
   0: The IWDG is not activated
   1: The IWDG is activated and needs to be refreshed regularly by the application

Bits 7:4  Reserved, must be kept at reset value.

Bit 3  EWU: Watchdog interrupt comparator value update
   This bit is set by hardware to indicate that an update of the interrupt comparator value
   (EWIT[11:0]) or an update of the EWIE is ongoing. It is reset by hardware when the update
   operation is completed in the VDD voltage domain (takes up to one period of presc_ck and
   two periods of the IWDG kernel clock iwdg_ker_ck).
   The EWIT[11:0] and EWIE fields can be updated only when EWU bit is reset.

Bit 2  WVU: Watchdog counter window value update
   This bit is set by hardware to indicate that an update of the window value is ongoing. It is
   reset by hardware when the reload value update operation is completed in the VDD voltage
   domain (takes up to one period of presc_ck and two periods of the IWDG kernel clock
   iwdg_ker_ck).
   The window value can be updated only when WVU bit is reset.
   This bit is generated only if generic “window” = 1.

Bit 1  RVU: Watchdog counter reload value update
   This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset
   by hardware when the reload value update operation is completed in the VDD voltage domain
   (takes up to six periods of the IWDG kernel clock iwdg_ker_ck).
   The reload value can be updated only when RVU bit is reset.

Bit 0  PVU: Watchdog prescaler value update
   This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is
   reset by hardware when the prescaler update operation is completed in the VDD voltage domain
   (takes up to six periods of the IWDG kernel clock iwdg_ker_ck).
   The prescaler value can be updated only when PVU bit is reset.

Note:  If several reload, prescaler, early interrupt position or window values are used by the
   application, it is mandatory to wait until RVU bit is reset before changing the reload value, to
   wait until PVU bit is reset before changing the prescaler value, to wait until WVU bit is reset.
before changing the window value, and to wait until EWU bit is reset before changing the early interrupt position value. After updating the prescaler and/or the reload/window/early interrupt value, it is not necessary to wait until RVU or PVU or WVU or EWU is reset before continuing code execution, except in case of low-power mode entry.

44.7.5 IWDG window register (IWDG_WINR)

Address offset: 0x10
Reset value: 0x0000 0FFF

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 31| 30| 29| 28| 27| 26| 25| 24| 23| 22| 21| 20| 19| 18| 17| 16|   |   |   |   |   |   |   |   |   |   |   |   |
| 15| 14| 13| 12| 11| 10|  9|  8|  7|  6|  5|  4|  3|  2|  1|  0|   |   |   |   |   |   |   |   |   |   |   |   |

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see Section 44.4.6. They contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the IWDCNT downcounter must be reloaded when its value is lower than WIN[11:0] + 1 and greater than 1.

The WVU bit in the **IWDG status register (IWDG_SR)** must be reset to be able to change the reload value.

**Note:** Reading this register returns the reload value from the VDD voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG status register (IWDG_SR) is reset.

44.7.6 IWDG early wake-up interrupt register (IWDG_EWCR)

Address offset: 0x14
Reset value: 0x0000 0000

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 31| 30| 29| 28| 27| 26| 25| 24| 23| 22| 21| 20| 19| 18| 17| 16|   |   |   |   |   |   |   |   |   |   |   |   |
| 15| 14| 13| 12| 11| 10|  9|  8|  7|  6|  5|  4|  3|  2|  1|  0|   |   |   |   |   |   |   |   |   |   |   |   |

<table>
<thead>
<tr>
<th>EWIE</th>
<th>EWIC</th>
<th>Res</th>
<th>Res</th>
<th>EWIT[11:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>w</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.
Bit 15 **EWIE**: Watchdog early interrupt enable  
Set and reset by software.  
0: The early interrupt interface is disabled.  
1: The early interrupt interface is enabled.  
The EWU bit in the **IWDG status register (IWDG_SR)** must be reset to be able to change the value of this bit.

Bit 14 **EWIC**: Watchdog early interrupt acknowledge  
The software must write a 1 into this bit in order to acknowledge the early wake-up interrupt and to clear the EWIF flag. Writing 0 has no effect, reading this flag returns a 0.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:0 **EWIT[11:0]**: Watchdog counter window value  
These bits are write access protected (see **Section 44.4.6**). They are written by software to define at which position of the IWDCNT down-counter the early wake-up interrupt must be generated. The early interrupt is generated when the IWDCNT is lower or equal to EWIT[11:0] - 1.  
EWIT[11:0] must be bigger than 1.  
An interrupt is generated only if EWIE = 1.  
The EWU bit in the **IWDG status register (IWDG_SR)** must be reset to be able to change the reload value.  
**Note**: Reading this register returns the Early wake-up comparator value and the Interrupt enable bit from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the EWU bit in the **IWDG status register (IWDG_SR)** is reset.
### 44.7.7  IWDG register map

#### Table 471. IWDG register map and reset values

| Offset | Register name | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | IWDG_KR       |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x04   | IWDG_PR       |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | IWDG_RLR      |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value |    |    | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x0C   | IWDG_SR       |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value |    |    | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10   | IWDG_WINR     |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value |    |    | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x14   | IWDG_EWCR     |          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3 on page 111 for the register boundary addresses.
45 System window watchdog (WWDG)

45.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence.

The watchdog circuit generates a reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit is cleared. A reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications requiring the watchdog to react within an accurate timing window.

45.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see Figure 616)
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

45.3 WWDG implementation

<table>
<thead>
<tr>
<th>WWDG mode / feature</th>
<th>WWDG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window function</td>
<td>X</td>
</tr>
<tr>
<td>Early wake-up interrupt generation</td>
<td>X</td>
</tr>
<tr>
<td>System reset generation&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>X</td>
</tr>
<tr>
<td>Capability to work in system Stop</td>
<td>-</td>
</tr>
<tr>
<td>Capability to work in system Standby</td>
<td>-</td>
</tr>
<tr>
<td>Capability to be frozen when the microcontroller enters in Debug mode&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>X</td>
</tr>
<tr>
<td>Option bytes to control the Hardware mode&lt;sup&gt;(4)&lt;/sup&gt;</td>
<td>X</td>
</tr>
</tbody>
</table>

1. "X" = supported, "-" = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via DBG_WWDG_STOP in DBG block.
45.4 **WWDG functional description**

If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent a reset. This operation can take place only when the counter value is lower than or equal to the window register value, and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to Figure 615 for the WWDG block diagram.

45.4.1 **WWDG block diagram**

![WWDG Block Diagram](image)

45.4.2 **WWDG internal signals**

*Table 473* gives the list of WWDG internal signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pclk</td>
<td>Digital input</td>
<td>APB bus clock</td>
</tr>
<tr>
<td>wwdg_out_rst</td>
<td>Digital output</td>
<td>WWDG reset signal output</td>
</tr>
<tr>
<td>wwdg_it</td>
<td>Digital output</td>
<td>WWDG early interrupt output</td>
</tr>
</tbody>
</table>
45.4.3 Enabling the watchdog

When the user option WWDG_SW selects “Software window watchdog”, the watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again, except by a reset.

When the user option WWDG_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

45.4.4 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value, due to the unknown status of the prescaler when writing to the WWDG_CR register (see Figure 616). The WWDG configuration register (WWDG_CFR) contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than or equal to the window register value, and greater than 0x3F. Figure 616 describes the window watchdog process.

Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

45.4.5 How to program the watchdog timeout

Use the formula in Figure 616 to calculate the WWDG timeout.

---

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.
The formula to calculate the timeout value is given by:

\[ t_{\text{WWDG}} = t_{\text{PCLK}} \times 4096 \times 2^{\text{WDGTB}[2:0]} \times (T[5:0] + 1) \] (ms)

where:
- \( t_{\text{WWDG}} \): WWDG timeout
- \( t_{\text{PCLK}} \): APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3 and T[5:0] is set to 63:

\[ t_{\text{WWDG}} = \frac{1}{48000} \times 4096 \times 2^3 \times (63 + 1) = 43.69\text{ms} \]

Refer to the datasheet for the minimum and maximum values of \( t_{\text{WWDG}} \).

**45.4.6 Debug mode**

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details, refer to *Section 58: Debug support (DBG)*.
45.5 **WWDG interrupts**

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

- Write EWIF bit of WWDG_SR register to 0, to clear unwanted pending interrupt
- Write EWI bit of WWDG_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog interrupt is generated, and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case the corresponding ISR must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

*Note:* When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.

<table>
<thead>
<tr>
<th>Table 474. WWDG interrupt requests</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Interrupt</strong></td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>WWDG(2)</td>
</tr>
</tbody>
</table>

1. The WWDG interrupt can have additional capabilities, refer to *Section 45.3* for details.
2. WWDG vector corresponds to the assertion of the wwdg_it signal.

45.6 **WWDG registers**

Refer to *Section 1.2 on page 101* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

45.6.1 **WWDG control register (WWDG_CR)**

Address offset: 0x000

Reset value: 0x0000 007F

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDGA</td>
</tr>
<tr>
<td>rs</td>
</tr>
</tbody>
</table>
Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WDGA**: Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled
1: Watchdog enabled

Bits 6:0 **T[6:0]**: 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every \((4096 \times 2^{WDGTB[2:0]})\) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

### 45.6.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>WDGTB[2:0]</td>
<td>EWI</td>
<td>W[6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rs</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WDGTB[2:0]**: Timer base

The timebase of the prescaler can be modified as follows:

000: CK counter clock (PCLK div 4096) div 1
001: CK counter clock (PCLK div 4096) div 2
010: CK counter clock (PCLK div 4096) div 4
011: CK counter clock (PCLK div 4096) div 8
100: CK counter clock (PCLK div 4096) div 16
101: CK counter clock (PCLK div 4096) div 32
110: CK counter clock (PCLK div 4096) div 64
111: CK counter clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI**: Early wake-up interrupt enable

Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]**: 7-bit window value

These bits contain the window value to be compared with the down-counter.
45.6.3  WWDG status register (WWDG_SR)

Address offset: 0x008
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:1  Reserved, must be kept at reset value.

Bit 0  EWIF: Early wake-up interrupt flag
This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

45.6.4  WWDG register map

The following table gives the WWDG register map and reset values.

Table 475. WWDG register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>WWDG_CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>WWDG_CFR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>WWDG_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 111 for the register boundary addresses.
46  Real-time clock (RTC)

46.1  Introduction

The RTC provides an automatic wake-up to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

The RTC is functional in V_{BAT} mode.

46.2  RTC main features

The RTC supports the following features (see Figure 617: RTC block diagram):

• Calendar with subseconds, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
• Binary mode with 32-bit free-running counter.
• Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
• Two programmable alarms.
• On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
• Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
• Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
• Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event, or by a switch to V_{BAT} mode.
• 17-bit auto-reload wake-up timer (WUT) for periodic events with programmable resolution and period.
• TrustZone support:
  – RTC fully securable
  – Alarm A, alarm B, wake-up Timer and timestamp individual secure or nonsecure configuration
• Alarm A, alarm B, wake-up Timer and timestamp individual privilege protection

The RTC is supplied through a switch that takes power either from the V_{DD} supply when present or from the V_{BAT} pin.

The RTC is functional in V_{BAT} mode and in all low-power modes when it is clocked by the LSE.

All RTC events (Alarm, wake-up Timer, Timestamp) can generate an interrupt and wake-up the device from the low-power modes.
46.3 RTC functional description

46.3.1 RTC block diagram
Figure 617. RTC block diagram

- **RTC_TS**
  - RTC_TR, RTC_DR
  - RTC_CALR
  - RTC_PRER
  - Smooth calibration
  - Asynchronous prescaler (default = 128)

- **WUCKSEL[1:0]**
  - Prescaler 2, 4, 8, 16

- **ck_apre**
  - (default 256 Hz)

- **RTC_WUTR**
  - 16-bit wake-up auto reload timer
  - ck_wut clock domain

- **RTC_TAMP**, rtc_tamp_evt
  - TAMPALRM
  - ALARM
  - TAMPOE
  - Smooth calibration

- **RTC_ALRMAR**
  - RTC_ALRMBR
  - RTC_ALRMASSR
  - RTC_ALRABINR

- **RTC_ALRBF**
  - WUTF
  - ALRAF
  - 0

- **RTC_LBER**
  - RTC_LBERBSSR
  - RTC_ALRBBINR

- **RTC_REFIN**
  - RTC_TS

- **RTC_TS**
  - Time stamp registers
  - RTC_TSTR
  - RTC_TSDR
  - RTC_TSSR

- **RTC_OUT1**, RTC_OUT2
  - Output control
  - TAMPALRM

- **rtc_ker_ck**
  - ck_spre (default 1 Hz)
  - Calendar
  - Shadow registers RTC_SSR

- **rtc_wut_trg**
  - rtc_calovf

- **rtc_alra_trg**
  - rtc_alrb_trg

- **rtc_pclk**
  - IRQ interface
  - Registers interface

- **rtc_tzen**

---

**Prescaler**
- 2, 4, 8, 16

**Wucksel[1:0]**
- Asynchronous prescaler (default = 128)

**ck_apre**
- (default 256 Hz)

**RTC_WUTR**
- 16-bit wake-up auto reload timer
- ck_wut clock domain

**RTC_TAMP**
- TAMPALRM
- ALARM
- TAMPOE
- Smooth calibration

**RTC_ALRMAR**
- RTC_ALRMBR
- RTC_ALRMASSR
- RTC_ALRABINR

**RTC_ALRBF**
- WUTF
- ALRAF
- 0

**RTC_LBER**
- RTC_LBERBSSR
- RTC_ALRBBINR

**RTC_REFIN**
- RTC_TS

**RTC_TS**
- Time stamp registers
- RTC_TSTR
- RTC_TSDR
- RTC_TSSR

**RTC_OUT1**, RTC_OUT2
- Output control
- TAMPALRM

** rtc_ker_ck**
- ck_spre (default 1 Hz)
- Calendar
- Shadow registers RTC_SSR

**rtc_wut_trg**
- rtc_calovf

**rtc_alra_trg**
- rtc_alrb_trg

**rtc_pclk**
- IRQ interface
- Registers interface

**rtc_tzen**

---

**RTC**
- Real-time clock
- Alarm A
  - RTC_ALRMAR
  - RTC_ALRMASSR
  - RTC_ALRABINR

- Alarm B
  - RTC_ALRMBR
  - RTC_ALRM8SSR
  - RTC_ALR8BINR

- Time stamp detection
- Time stamp registers
- RTC_TSTR
- RTC_TSDR
- RTC_TSSR

- Output control
- TAMPALRM
- ALARM
- TAMPOE

- Smooth calibration

- Asynchronous prescaler (default = 128)

- Synchronous prescaler (default = 256)

- Smooth calibration

- Asynchronous prescaler (default = 128)

- Synchronous prescaler (default = 256)

- Smooth calibration
46.3.2 RTC pins and internal signals

Table 476. RTC input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_TS</td>
<td>Input</td>
<td>RTC timestamp input</td>
</tr>
<tr>
<td>RTC_REFIN</td>
<td>Input</td>
<td>RTC 50 or 60 Hz reference clock input</td>
</tr>
<tr>
<td>RTC_OUT1</td>
<td>Output</td>
<td>RTC output 1</td>
</tr>
<tr>
<td>RTC_OUT2</td>
<td>Output</td>
<td>RTC output 2</td>
</tr>
</tbody>
</table>

RTC_OUT1 and RTC_OUT2 which select one of the following two outputs:
- CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
- TAMPALRM: This output is the OR between rtc_tamp_evt and ALARM signals.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wake-up outputs. rtc_tamp_evt is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 477. RTC internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtc_ker_ck</td>
<td>Input</td>
<td>RTC kernel clock, also named RTCCLK in this document</td>
</tr>
<tr>
<td>rtc_pclk</td>
<td>Input</td>
<td>RTC APB clock</td>
</tr>
<tr>
<td>rtc_its</td>
<td>Input</td>
<td>RTC internal timestamp event</td>
</tr>
<tr>
<td>rtc_tamp_evt</td>
<td>Input</td>
<td>Tamper event (internal or external) detected in TAMP peripheral</td>
</tr>
<tr>
<td>rtc_tzen</td>
<td>Input</td>
<td>RTC TrustZone enabled</td>
</tr>
<tr>
<td>rtc_it</td>
<td>Output</td>
<td>RTC interrupts (refer to Section 46.5: RTC interrupts for details)</td>
</tr>
<tr>
<td>rtc_alra_trg</td>
<td>Output</td>
<td>RTC alarm A event detection trigger</td>
</tr>
<tr>
<td>rtc_alrb_trg</td>
<td>Output</td>
<td>RTC alarm B event detection trigger</td>
</tr>
<tr>
<td>rtc_wut_trg</td>
<td>Output</td>
<td>RTC wake-up timer event detection trigger</td>
</tr>
<tr>
<td>rtc_calovf</td>
<td>Output</td>
<td>RTC calendar overflow: this signal is generated when the RTC calendar reaches its maximum value, on the 31st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.</td>
</tr>
</tbody>
</table>

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes or VBAT when the selected clock is not LSE. Refer to Section 46.4: RTC low-power modes for more details.
The TZEN option bit is used to activate TrustZone in the device.
TZEN = 1: TrustZone activated.
TZEN = 0: TrustZone disabled.
When TrustZone is disabled, the APB access to the RTC registers are nonsecure.
The triggers outputs can be used as triggers for other peripherals.

### 46.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the Battery Backup Domain (Vs) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

RTC_OUT1, RTC_TS, TAMP_IN1, TAMP_OUT2 and TAMP_OUT3 are mapped on the same pin (PC13). The RTC and TAMP functions mapped on PC13 are available in all low-power modes and in VBAT mode.

The output mechanism follows the priority order shown in **Table 479**.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source/destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtc_its</td>
<td>From power controller (PWR): main power loss/switch to VBAT detection output</td>
</tr>
<tr>
<td>rtc_tamp_evt</td>
<td>From TAMP peripheral: tamp_evt</td>
</tr>
<tr>
<td>rtc_tzen</td>
<td>From FLASH option bytes: TZEN</td>
</tr>
<tr>
<td>rtc_calovf</td>
<td>To TAMP peripheral: tamp_itamp5</td>
</tr>
<tr>
<td>PC13 Pin function</td>
<td>OSEL[1:0] (ALARM output enable)</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>TAMPA LRM output</td>
<td>01 or 10 or 11</td>
</tr>
<tr>
<td>Push-Pull</td>
<td>00</td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
</tr>
<tr>
<td>TAMPA LRM output</td>
<td>01 or 10 or 11</td>
</tr>
<tr>
<td>Open-Drain(2)</td>
<td>00</td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
</tr>
<tr>
<td>Internal pull-up</td>
<td>01 or 10 or 11</td>
</tr>
<tr>
<td></td>
<td>00</td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
</tr>
<tr>
<td>CALIB output PP</td>
<td>00</td>
</tr>
<tr>
<td>TAMP_OUT2 output</td>
<td>00</td>
</tr>
<tr>
<td>PP</td>
<td>00</td>
</tr>
<tr>
<td>TAMP_OUT3 output</td>
<td>00</td>
</tr>
</tbody>
</table>
## Table 479. RTC pin PC13 configuration\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>PC13 Pin function</th>
<th>OSEL([1:0]) (ALARM output enable)</th>
<th>TAMPOE (TAMPER output enable)</th>
<th>COE (CALIB output enable)</th>
<th>OUT2EN</th>
<th>TAMPALRM_TYPE</th>
<th>TAMPAE=TAMPAE=1 with ATOSHARE=0, or TAMPAE=1 with ATOSHARE=1, and ATOSELx=1</th>
<th>TAMPAE=TAMPAE=1 with ATOSHARE=0, or TAMPAE=1 with ATOSHARE=1, and ATOSELx=2 and OUT3_RMP=00</th>
<th>TAMPAE=TAMPAE=1 with ATOSHARE=0, or TAMPAE=1 with ATOSHARE=1, and ATOSELx=2 and OUT3_RMP=00</th>
<th>TAMP1E (TAMP_IN1 input enable)</th>
<th>TSE (RTC_TS input enable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMP_IN1 input floating</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>Don't care</td>
<td>Don't care</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTC_TS and TAMP_IN1 input floating</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTC_TS input floating</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wakeup pin</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard GPIO</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. OD: open drain; PP: push-pull.
2. In this configuration the GPIO must be configured in input.
RTC_OUT2, TAMP_IN2 and TAMP_OUT3 are mapped on the same pin (PI8). PI8 configuration is controlled by the RTC, whatever the PI8 GPIO configuration. The RTC or TAMP functions mapped on PI8 are available in all low-power modes and in VBAT mode.

The output mechanism follows the priority order shown in Table 480.

<table>
<thead>
<tr>
<th>PI1 Pin function</th>
<th>OSEL[1:0] (ALARM output enable)</th>
<th>TAMPOE (TAMPER output enable)</th>
<th>COE (CALIB output enable)</th>
<th>OUT2EN &amp; OUT2_RMP</th>
<th>TAMPALRM_TYPE</th>
<th>TAMPALRM_PU</th>
<th>TAMP3E=TAMP3AM=1 with ATOSHARE=0 and OUT3_RMP=01, TAMPx=E=1 with ATOSHARE=1, TAMPOE=x and OUT2_RMP=01, TAMPOE=x and OUT2_RMP=01</th>
<th>TAMP2E &amp; IN2_RMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMPALRM output</td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>Push-Pull</td>
<td>00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAMPALRM output</td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>Open-Drain</td>
<td>00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal pull-up</td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALIB output PP</td>
<td>00</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>TAMP_OUT3 output</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAMP_IN2 input</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>floating</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wakeup pin</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Standard GPIO</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
In addition, it is possible to output RTC_OUT2 on PB2 or PI8 pin thanks to OUT2EN and OUT2_RMP bits. The different functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, COE and OUT2EN configuration, as shown in table Table 481.

### Table 481. RTC_OUT mapping

<table>
<thead>
<tr>
<th>OSEL[1:0] bits ALARM output enable</th>
<th>COE bit (CALIB output enable)</th>
<th>OUT2EN bit</th>
<th>RTC_OUT1 on PC13</th>
<th>RTC_OUT2 on PB2 or PI8</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>-CALIB</td>
<td>-</td>
</tr>
<tr>
<td>01 or 10 or 11</td>
<td>Don’t care</td>
<td>0</td>
<td>TAMPALRM</td>
<td>-</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>TAMPALRM</td>
<td>CALIB</td>
</tr>
<tr>
<td>01 or 10 or 11</td>
<td>0</td>
<td>1</td>
<td>TAMPALRM</td>
<td>CALIB</td>
</tr>
</tbody>
</table>

#### 46.3.4 RTC secure protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both secure and nonsecure modes, except for the RTC secure configuration register (RTC_SECCFGR) which can be written in secure mode only. The RTC protection configuration is not affected by a system reset.

When the SEC bit is set in the RTC_SECCFGR register:
- Writing the RTC registers is possible only in secure mode.
- Reading RTC_SECCFGR, RTC_PRIVCFGR, RTC_MISR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in secure and nonsecure modes. All the other RTC registers can be read only in secure mode.

When the SEC bit is cleared, it is still possible to protect some of the registers by setting dedicated INITSEC, CALSEC, TSSEC, WUTSEC, ALRASEC or ALRBSEC control bits. If all these bits are also set, all the RTC registers can be read in secure and nonsecure mode.

- When INITSEC is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR, FMT control bits in RTC_CR and INITPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and nonsecure mode.

- When CALSEC is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR and CALPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and nonsecure mode.

- When ALRASEC is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE, ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, CALRAF and CSSRUF in the
RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_SMISR can be read and written only in secure mode.

- ALRAPRIV in the RTC_PRIVCFGR can be written only in secure mode.

- When ALRBSEC is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE ALRBFCLR, ALRBIE in the RTC_CR, CALRBF in the RTC_SCR, ALRBF in RTC_SR, and ALRBMF in RTC_SMISR can be read and written only in secure mode.
  - ALRBPRIV in the RTC_PRIVCFGR can be written only in secure mode.

- When WUTSEC is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_SMISR can be read and written only in secure mode.
  - WUTPRIV in the RTC_PRIVCFGR can be written only in secure mode.

- When TSSEC is set:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_SMISR can be read and written only in secure mode.
  - TSPRIV in the RTC_PRIVCFGR can be written only in secure mode.

A nonsecure access to a secure-protected register is denied:

- There is no bus error generated.
- In case the register has a global protection: a notification is generated through a flag/interrupt in the TZIC (TrustZone illegal access controller). In case only a few bits of the register are protected (for registers with mixed features such as RTC_CR...), no notification is generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the RTC reset and clock control is also secured in the RCC.

### 46.3.5 RTC privilege protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both privileged and non-privileged modes, except for the RTC privilege mode control register (RTC_PRIVCFGR) which can be written in privilege mode only. The RTC protection configuration is not affected by a system reset.

When the PRIV bit is set in the RTC_PRIVCFGR register:

- Writing the RTC registers is possible only in privileged mode.
- Reading the RTC_SECCFG, RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privilege and non-privileged modes. All the other RTC registers can be read only in privilege mode.

When the PRIV bit is cleared, it is still possible to protect some of the registers by setting dedicated INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRIV or ALRBPRIV control bits. If
all these bits are also cleared, all the RTC registers can be read or written in privilege and non-privileged modes.

- When INITPRIV is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR and FMT control bits in RTC_CR, plus INITSEC in the RTC_SECCFGR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and non-privileged mode.

- When CALPRIV is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR, plus CALDSEC in the RTC_SECCFGR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and non-privileged mode.

- When ALRAPRIV is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, and CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - ALRAFSEC in the RTC_SECCFGR can be written only in privilege mode.

- When ALRBPRIV is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE ALRBFCLR, ALRBIE in the RTC_CR, and CALRBF in the RTC_SCR, ALRBF in RTC_SR, and ALRBMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - ALRBSEC in the RTC_SECCFGR can be written only in privilege mode.

- When WUTPRIV is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, and CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - WUTSEC in the RTC_SECCFGR can be written only in privilege mode.

- When TSPRIV is set:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - TSSEC in the RTC_SECCFGR can be written only in privilege mode.

A non-privileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

46.3.6 Clock and prescalers

For more information on the RTC clock (RTCCLK) source configuration, refer to “Reset and clock control (RCC)”.
BCD mode (BIN=00)

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see Figure 617: RTC block diagram):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

**Note:** When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is $2^{22}$.

This corresponds to a maximum input frequency of around 4 MHz.

The $f_{ck\_apre}$ is given by the following formula:

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

The $ck\_apre$ clock is used to clock the binary RTC_SSR subsecond downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

The $f_{ck\_spre}$ is given by the following formula:

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

**Binary mode (BIN=01)**

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are not functional.

This down-counter is clocked by $ck\_apre$: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.

PREDIV_S value is don't care.

**Mixed mode (BIN=10 or 11)**

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are also available.

This down-counter is clocked by $ck\_apre$: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register. The bits BCDU[2:0] are
used to define when the calendar is incremented by 1 second, using the SSR least significant bits.

46.3.7 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see Section 46.6.12: RTC shift control register (RTC_SHIFTR)). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock (fAPB) must be at least 7 times the frequency of the RTC clock (fRTCCLK).

The shadow registers are reset by system reset.

46.3.8 Calendar ultra-low power mode

It is possible to reduce drastically the RTC power consumption by setting the LPCAL bit in the RTC_CALR register. In this configuration, the whole RTC is clocked by ck_apre only instead of both RTCCLK and ck_apre. Consequently, some flags delays are longer, and the calibration window is longer (refer to Section : RTC ultra-low-power mode).

The LPCAL bit is ignored (assumed to be 0) when asynchronous prescaler division factor (PREDIV_A+1) is not a power of 2.

Switching from LPCAL=0 to LPCAL=1 or from LPCAL=1 to LPCAL=0 is not immediate and requires a few ck_apre periods to complete.

46.3.9 Programmable alarms

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.
When the binary mode is used, the subsecond field can be programmed in the alarm binary register RTC_ALRABINR.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

In case the Alarm is used to generate a trigger event for another peripheral, the ALRAF can be automatically cleared by hardware by configuring the ALRAFCLR bit at 1 in the RTC_CR register. In this configuration there is no need for software intervention if the only purpose is clearing the ALRAF flag.

**Caution:**

If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL the RTC_CR register.

### 46.3.10 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC_CR register.

The wake-up timer clock input ck_wut can be:

- **RTC clock (RTCCLK) divided by 2, 4, 8, or 16.**
  - When RTCCLK is LSE (32.768 kHz), this permits the wake-up interrupt period to be configured from 122 µs to 32 s, with a resolution down to 61 µs.
- **ck_spre (usually 1 Hz internal clock) in BCD mode, or the clock used to update the calendar as defined by BCDU in binary or mixed (BCD-binary) modes.**
  - When ck_spre frequency is 1 Hz, a wake-up time from 1 s to around 36 hours can be achieved with one-second resolution. This large programmable time range is divided in 2 parts:
    - from 1 s to 18 hours when WUCKSEL [2:1] = 10
    - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case $2^{16}$ is added to the 16-bit counter current value. When the initialization sequence is complete (see *Programming the wake-up timer on page 1963*), the timer starts counting down. When the wake-up function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wake-up counter is automatically reloaded with its reload value (RTC_WUTR register value).

Depending on WUTOCLR in the RTC_WUTR register, the WUTF flag must either be cleared by software (WUTOCLR = 0x0000), or the WUTF is automatically cleared by hardware when the auto-reload down counter reaches WUTOCLR value (0x0000 < WUTOCLR ≤ WUT).

The wake-up flag is output on an internal signal rtc_wut that can be used by other peripherals (refer to section *Section 46.3.1: RTC block diagram*).

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.
The periodic wake-up flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop, and Standby) have no influence on the wake-up timer.

### 46.3.11 RTC initialization and configuration

**RTC Binary, BCD or Mixed mode**

By default the RTC is in BCD mode (BIN = 00 in the RTC_ICSR register): the RTC_SSR register contains the subsecond field SS[15:0], clocked by ck_apre, allowing to generate a 1 Hz clock to update the calendar registers in BCD format (RTC_TR and RTC_DR).

When the RTC is configured in binary mode (BIN = 01 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar registers in BCD format (RTC_TR and RTC_DR) are not used.

When the RTC is configured in mixed mode (BIN = 10 or 11 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar is updated (1 second increment) each time the SSR[BCDU+7:0] reaches 0.

**RTC register write protection**

After system reset, the RTC registers are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, some of the RTC registers are write-protected: RTC_TR, RTC_DR, RTC_PRER, RTC_CALR, RTC_SHIFTR, the bits INIT, BIN and BCDU in RTC_ICSR and the bits FMT, SUB1H, ADD1H, REFCKON in RTC_CR.

The following steps are required to unlock the write protection on the protected RTC registers.

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

The registers protected by INITPRIV are write-protected by the INIT KEY.

The registers protected by CALDPRIV are write-protected by the CAL KEY.

In case PRIV or INITPRIV is set in the RTC_PRIVCFGR, and/or SEC or INITSEC is set in the RTC_SECCFGR: the INIT KEY is unlocked and locked only if the write accesses into the RTC_WPR register are done in the privilege and security mode defined by PRIV, INITPRIV, SEC, INITSEC configuration.

In case PRIV or CALPRIV is set in the RTC_PRIVCFGR, and/or SEC or CALSEC is set in the RTC_SECCFGR: the CAL KEY is unlocked and locked only if the write accesses into the RTC_WPR register are done in the privilege and security mode defined by PRIV, CALPRIV, SEC, CALSEC configuration.
Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.

2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1.
   - If LPCAL=0: INITF is set around 2 RTCCLK cycles after INIT bit is set.
   - If LPCAL=1: INITF is set up to 2 ck_apre cycle after INIT bit is set.

3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register, plus BIN and BCDU in the RTC_ICSR register.

4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.

5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded.
   - If LPCAL=0: the counting restarts after 4 RTCCLK clock cycles.
   - If LPCAL=1: the counting restarts after up to 2 RTCCLK + 1 ck_apre.

When the initialization sequence is complete, the calendar starts counting. The RTC_SSR content is initialized with:
- PREDIV_S in BCD mode (BIN=00)
- 0xFFFF FFFF in binary or mixed (BCD-binary) modes (BIN=01, 10 or 11).

In BCD mode, RTC_SSR contains the value of the synchronous prescaler counter. This enables one to calculate the exact time being maintained by the RTC down to a resolution of \(1 / (PREDIV_S + 1)\) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value \(PREDIV_S[14:0]\). The maximum resolution allowed (30.52 \(\mu\)s with a 32768 Hz clock) is obtained with \(PREDIV_S\) set to 0x7FFF.

However, increasing \(PREDIV_S\) means that \(PREDIV_A\) must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption. The RTC dynamic consumption is optimized for \(PREDIV_A+1\) being a power of 2.

Note: After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00).

Note: To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.
Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRABINR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wake-up timer

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wake-up timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wake-up auto-reload counter and to WUCKSEL[2:0] bits is allowed. This step must be skipped in calendar initialization mode.
   - If WUCKSEL[2] = 0: WUTWF is set around 1ck_wut + 1 RTCCLK cycles after WUTE bit is cleared.
   - If WUCKSEL[2] = 1: WUTWF is set up to 1ck_apre + 1 RTCCLK cycles after WUTE bit is cleared.
3. Program the wake-up auto-reload value WUT[15:0], WUTOCLR[15:0] and the wake-up clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wake-up timer restarts down-counting.
   - If WUCKSEL[2] = 0: WUTWF is cleared around 1ck_wut + 1 RTCCLK cycles after WUTE bit is set.
   - If WUCKSEL[2] = 1: WUTWF is cleared up to 1ck_apre + 1 RTCCLK cycles after WUTE bit is set.

46.3.12 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB clock frequency (fPCLK) must be equal to or greater than seven times the RTC clock frequency (fRTCCLK). This ensures a secure behavior of the synchronization mechanism.

If the APB clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycle. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and
then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wake-up and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to Calendar initialization and configuration on page 1962): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to Section 46.3.14: RTC synchronization): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLOCK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLOCK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

46.3.13 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the wake-up timer register (RTC_WUTR), the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR/RTC_ALRABINR and RTC_ALRMBSSR/RTC_ALRMBR/RTC_ALRBBINR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to RCC for details about RTC clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.
46.3.14 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the subsecond field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by "shifting" its clock by a fraction of a second using RTC_SHIFTR.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock with a resolution of 1 ck_apre period.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock.

If at the same time the ADD1S bit is set in BCD or mixed mode, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock. ADD1S has no effect in binary mode.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: In mixed mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Caution: Before initiating a shift operation in BCD mode, the user must check that SS[15] = 0 in order to ensure that no overflow occurs. In mixed mode, the user must check that the bit SS[BCDU+8] = 0.

Caution: This synchronization feature is not compatible with the reference clock detection feature; firmware must not write to RTC_SHIFTR when REFCKON = 1.

46.3.15 RTC reference clock detection

This feature is available only in BCD mode (BIN=00).

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same
moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREDIV_S = 0x00FF

**Note:** RTC_REFIN clock detection is not available in Standby mode.

### 46.3.16 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual ck_cal pulses).

If LPCAL=0: ck_cal = RTCCLK

If LPCAL=1: ck_cal = ck_apre

These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

**RTC ultra-low-power mode**

The RTC consumption can be reduced by setting the LPCAL bit in the RTC calibration register (RTC_CALR). In this case, the calibration mechanism is applied on ck_apre instead of RTCCLK. The resulting accuracy is the same, but the calibration is performed during a calibration cycle of about $2^{20} \times$ PREDIV_A x RTCCLK pulses instead of $2^{20}$ RTCCLK pulses when LPCAL=0.

**Smooth calibration mechanism**

The smooth calibration register (RTC_CALR) specifies the number of ck_cal clock cycles to be masked during the calibration cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked.
- Setting CALM[2] to 1 causes four additional cycles to be masked.
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

**Note:** CALM[8:0] (RTC_CALR) specifies the number of ck_cal pulses to be masked during the calibration cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM permits the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP
to 1 effectively inserts an extra \textit{ck\textsubscript{cal}} pulse every $2^{11}$ \textit{ck\textsubscript{cal}} cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 \textit{ck\textsubscript{cal}} cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (\textit{FCAL}) given the input frequency (\textit{FRTCCLK}) is as follows:

$$\textit{FCAL} = \textit{FRTCCLK} \times \left[1 + (\text{CALP} \times 512 - \text{CALM}) / (2^{20} + \text{CALM} - \text{CALP} \times 512)\right]$$

\textbf{Caution:} \textit{PREDIV\_A} must be greater or equal to 3.

\textbf{Calibration when \textit{PREDIV\_A} < 3}

The CALP bit can not be set to 1 when the asynchronous prescaler value (\textit{PREDIV\_A} bits in \texttt{RTC\_PRER} register) is less than 3. If CALP was already set to 1 and \textit{PREDIV\_A} bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

It is however possible to perform a calibration with \textit{PREDIV\_A} less than 3 in BCD mode, the synchronous prescaler value (\textit{PREDIV\_S}) should be reduced so that each second is accelerated by 8 \textit{ck\textsubscript{cal}} clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when \textit{PREDIV\_A} equals 1 (division factor of 2), \textit{PREDIV\_S} should be set to 16379 rather than 16383 (4 less). The only other interesting case is when \textit{PREDIV\_A} equals 0, \textit{PREDIV\_S} should be set to 32759 rather than 32767 (8 less).

If \textit{PREDIV\_S} is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$\textit{FCAL} = \textit{FRTCCLK} \times \left[1 + (256 - \text{CALM}) / (2^{20} + \text{CALM} - 256)\right]$$

In this case, \texttt{CALM}[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

\textbf{Verifying the RTC calibration}

It is recommended to verify the RTC calibration with LPCAL = 0, in order to have a 32-second calibration cycle.

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.
However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.
  Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLOCK cycles over 32 seconds, due to the limitation of the calibration resolution).
- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16-second calibration cycle period.
  In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLOCK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.
- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.
  In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLOCK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

**Re-calibration on-the-fly**

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

**46.3.17 Timestamp function**

Timestamp is enabled by setting the TSE or ITSE bits of RTC_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal or external tamper event is detected. Refer to RTC control register (RTC_CR) and refer to Section : Timestamp on tamper event.

When ITSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the VBAT supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC_SR register is set. In case the event is internal, the ITSF flag is also set in RTC_SR register.
By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

**Note:** TSF is set up to 2 ck_apre cycles after the timestamp event from RTC_TS pin or from rtc_its internal signal occurs due to synchronization process. TSF is set up to 3 ck_apre cycles after tamper flags.

TSOVF is set up to only 1 ck_apre cycle after the event occurs. This means that if two timestamp events are close together, TSOVF can be seen as ‘1’ while TSF is still ‘0’. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

**Caution:** If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

### 46.3.18 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is fRTCCLK/64. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is fRTCCLK/(256 * (PREDIV_A+1)). This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = 0x7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

**Note:** When COSEL is cleared, the CALIB output is the output of the 6th stage of the asynchronous prescaler. If LPCAL is changed from 0 to 1, the output can be irregular (glitch…) during the LPCAL switch. If LPCAL = 1 this output is always available. If LPCAL = 0, no output is present if PREDIV_A is < 0x20.

When COSEL is set, the CALIB output is the output of the 8th stage of the synchronous prescaler.

### 46.3.19 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

When the TAMPOE control bit is set in the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wake-up flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.
TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

Note: Once the TAMPALRM output is enabled, it has priority over CALIB on RTC_OUT1. In case the TAMPALRM is configured open-drain in the RTC, the RTC_OUT1 GPIO must be configured as input.

46.4 RTC low-power modes

Table 482. Effect of low-power modes on RTC

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect&lt;br&gt;RTC interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts&lt;br&gt;cause the device to exit the Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts&lt;br&gt;cause the device to exit the Standby mode.</td>
</tr>
</tbody>
</table>

The table below summarizes the RTC pins and functions capability in all modes.

Table 483. RTC pins functionality over modes

<table>
<thead>
<tr>
<th>Functions</th>
<th>Functional in all low-power modes except Standby mode</th>
<th>Functional in Standby mode</th>
<th>Functional in V_{BAT} mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_TS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RTC_REFIN</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RTC_OUT1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RTC_OUT2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

46.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration. The nonsecure interrupt output or the secure interrupt output is also activated.
### Table 484. Nonsecure interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag(1)</th>
<th>Enable control bit(2)</th>
<th>Interrupt clear method</th>
<th>Exit from low-power modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC</td>
<td>Alarm A</td>
<td>ALRAF</td>
<td>ALRAIE and (ALRASEC=0 and SEC=0)</td>
<td>write 1 in CALRAF</td>
<td>Yes(3)</td>
</tr>
<tr>
<td></td>
<td>Alarm B</td>
<td>ALRBF</td>
<td>ALRBIE and (ALRBSEC=0 and SEC=0)</td>
<td>write 1 in CALRBF</td>
<td>Yes(3)</td>
</tr>
<tr>
<td></td>
<td>Timestamp</td>
<td>TSF</td>
<td>TSIE and (TSSEC=0 and SEC=0)</td>
<td>write 1 in CTSF</td>
<td>Yes(3)</td>
</tr>
<tr>
<td></td>
<td>Wake-up timer</td>
<td>WUTF</td>
<td>WUTIE and (WUTSEC=0 and SEC=0)</td>
<td>write 1 in CWUTF</td>
<td>Yes(3)</td>
</tr>
<tr>
<td></td>
<td>SSR underflow</td>
<td>SSRUF</td>
<td>SSRUIE and (ALRASEC=0 and SEC=0)</td>
<td>write 1 in CSSRUF</td>
<td>Yes(3)</td>
</tr>
</tbody>
</table>

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

### Table 485. Secure interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag(1)</th>
<th>Enable control bit(2)</th>
<th>Interrupt clear method</th>
<th>Exit from low-power modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_S</td>
<td>Alarm A</td>
<td>ALRAF</td>
<td>ALRAIE and (ALRASEC=1 or SEC=1)</td>
<td>write 1 in CALRAF</td>
<td>Yes(3)</td>
</tr>
<tr>
<td></td>
<td>Alarm B</td>
<td>ALRBF</td>
<td>ALRBIE and (ALRBSEC=1 or SEC=1)</td>
<td>write 1 in CALRBF</td>
<td>Yes(3)</td>
</tr>
<tr>
<td></td>
<td>Timestamp</td>
<td>TSF</td>
<td>TSIE and (TSSEC=1 or SEC=1)</td>
<td>write 1 in CTSF</td>
<td>Yes(3)</td>
</tr>
<tr>
<td></td>
<td>Wake-up timer</td>
<td>WUTF</td>
<td>WUTIE and (WUTSEC=1 or SEC=1)</td>
<td>write 1 in CWUTF</td>
<td>Yes(3)</td>
</tr>
<tr>
<td></td>
<td>SSR underflow</td>
<td>SSRUF</td>
<td>SSRUIE and (ALRASEC=1 or SEC=1)</td>
<td>write 1 in CSSRUF</td>
<td>Yes(3)</td>
</tr>
</tbody>
</table>

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_SMISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.
46.6 RTC registers

Refer to Section 1.2 of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

46.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to Calendar initialization and configuration on page 1962 and Reading the calendar on page 1963.

This register is write protected. The write access procedure is described in RTC register write protection on page 1961.

This register can be write-protected against nonsecure access. Refer to Section 46.3.4: RTC secure protection modes.

This register can be write-protected against non-privileged access. Refer to Section 46.3.5: RTC privilege protection modes.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

15  14  13  12  11  10  9  8  7  6  5  4  3  2  1  0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----</td>
<td>---------</td>
<td>---------</td>
<td>-----</td>
<td>--------</td>
<td>--------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format
46.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to Calendar initialization and configuration on page 1962 and Reading the calendar on page 1963.

This register is write protected. The write access procedure is described in RTC register write protection on page 1961.

This register can be write-protected against nonsecure access. Refer to Section 46.3.4: RTC secure protection modes.

This register can be write-protected against non-privileged access. Refer to Section 46.3.5: RTC privilege protection modes.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 YT[3:0]: Year tens in BCD format

Bits 19:16 YU[3:0]: Year units in BCD format

Bits 15:13 WDU[2:0]: Week day units

000: forbidden
001: Monday
...
111: Sunday

Bit 12 MT: Month tens in BCD format

Bits 11:8 MU[3:0]: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 DT[1:0]: Date tens in BCD format

Bits 3:0 DU[3:0]: Date units in BCD format

Note: The calendar is frozen when reaching the maximum value, and can't roll over.
46.6.3 RTC subsecond register (RTC_SSR)

Address offset: 0x08
Backup domain reset value: 0x0000 0000
System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **SS[31:0]**: Synchronous binary counter
- **SS[31:16]**: Synchronous binary counter MSB values
  - When Binary or Mixed mode is selected (BIN = 01 or 10 or 11):
    - SS[31:16] are the 16 MSB of the SS[31:0] free-running down-counter.
  - When BCD mode is selected (BIN=00):
    - SS[31:16] are forced by hardware to 0x0000.
- **SS[15:0]**: Subsecond value/synchronous binary counter LSB values
  - When Binary mode is selected (BIN = 01 or 10 or 11):
    - SS[15:0] are the 16 LSB of the SS[31:0] free-running down-counter.
  - When BCD mode is selected (BIN=00):
    - SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:
      - Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)
      - SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.
### 46.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in RTC register write protection on page 1961.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to Section 46.3.4: RTC secure protection modes.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to Section 46.3.5: RTC privilege protection modes.

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to 0

| Bit 31:17 | Reserved, must be kept at reset value. |
| Bit 16   | RECALPF: Recalibration pending Flag |
|          | The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to Re-calibration on-the-fly. |
| Bit 15:13| Reserved, must be kept at reset value. |
| Bits 12:10 | BCDU[2:0]: BCD update (BIN = 10 or 11) |
|          | In mixed mode when both BCD calendar and binary extended counter are used (BIN = 10 or 11), the calendar second is incremented using the SSR Least Significant Bits. |
|          | 0x0: 1s calendar increment is generated each time SS[7:0] = 0 |
|          | 0x1: 1s calendar increment is generated each time SS[8:0] = 0 |
|          | 0x2: 1s calendar increment is generated each time SS[9:0] = 0 |
|          | 0x3: 1s calendar increment is generated each time SS[10:0] = 0 |
|          | 0x4: 1s calendar increment is generated each time SS[11:0] = 0 |
|          | 0x5: 1s calendar increment is generated each time SS[12:0] = 0 |
|          | 0x6: 1s calendar increment is generated each time SS[13:0] = 0 |
|          | 0x7: 1s calendar increment is generated each time SS[14:0] = 0 |
| Bits 9:8 | BIN[1:0]: Binary mode |
|          | 00: Free running BCD calendar mode (Binary mode disabled). |
|          | 01: Free running Binary mode (BCD mode disabled) |
|          | 10: Free running BCD calendar and Binary modes |
|          | 11: Free running BCD calendar and Binary modes |
Bit 7  **INIT**: Initialization mode
0: Free running mode
1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER), plus BIN and BCDU fields. Counters are stopped and start counting from the new value when INIT is reset.

Bit 6  **INITF**: Initialization flag
When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.
0: Calendar registers update is not allowed
1: Calendar registers update is allowed

Bit 5  **RSF**: Registers synchronization flag
This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPShAD = 1). This bit can also be cleared by software. It is cleared either by software or by hardware in initialization mode.
0: Calendar shadow registers not yet synchronized
1: Calendar shadow registers synchronized

Bit 4  **INITS**: Initialization status flag
This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).
0: Calendar has not been initialized
1: Calendar has been initialized

Bit 3  **SHPF**: Shift operation pending
This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.
0: No shift operation is pending
1: A shift operation is pending

Bit 2  **WUTWF**: Wake-up timer write flag
This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.
It is cleared by hardware in initialization mode.
0: Wake-up timer configuration update not allowed except in initialization mode
1: Wake-up timer configuration update allowed

Bits 1:0  Reserved, must be kept at reset value.
46.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to Calendar initialization and configuration on page 1962.

This register is write protected. The write access procedure is described in RTC register write protection on page 1961.

This register can be write-protected against nonsecure access. Refer to Section 46.3.4: RTC secure protection modes.

This register can be write-protected against non-privileged access. Refer to Section 46.3.5: RTC privilege protection modes.

Address offset: 0x10
Backup domain reset value: 0x007F 00FF
System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREDIV_S[14:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 PREDIV_A[6:0]: Asynchronous prescaler factor
This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 PREDIV_S[14:0]: Synchronous prescaler factor
This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)
46.6.6 **RTC wake-up timer register (RTC_WUTR)**

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**WUTOCLR[15:0]**: Wake-up auto-reload output clear value

When WUTOCLR[15:0] is different from 0x0000, WUTF is set by hardware when the auto-reload down-counter reaches 0 and is cleared by hardware when the auto-reload downcounter reaches WUTOCLR[15:0]. When WUTOCLR[15:0] = 0x0000, WUTF is set by hardware when the WUT down-counter reaches 0 and is cleared by software.

**Bits 15:0** **WUT[15:0]**: Wake-up auto-reload value bits

When the wake-up timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.

When WUCKSEL[2] = 1, the wake-up timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs between WUT and (WUT + 2) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

46.6.7 **RTC control register (RTC_CR)**

*This register is write protected. The write access procedure is described in RTC register write protection on page 1961.*
This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to Section 46.3.4: RTC secure protection modes.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to Section 46.3.5: RTC privilege protection modes.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT2EN</td>
<td>TAMPALRM_TYPE</td>
<td>TAMPOE</td>
<td>ALRBFE</td>
<td>ALRBFCLR</td>
<td>TAMP</td>
<td>TSIE</td>
<td>IE</td>
<td>WUTIE</td>
<td>WUTE</td>
<td>TSEL</td>
<td>WUSE</td>
<td>TAMPALRM</td>
<td>RSS</td>
<td>BYP</td>
<td>SHAD</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31 **OUT2EN**: RTC_OUT2 output enable

With this bit set, the RTC outputs can be remapped on RTC_OUT2 as follows:

**OUT2EN = 0**: RTC output 2 disable

If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1

If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1

**OUT2EN = 1**: RTC output 2 enable

If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2

If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2

If (OSEL ≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.

Bit 30 **TAMPALRM_TYPE**: TAMPALRM output type

0: TAMPALRM is push-pull output

1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM PU**: TAMPALRM pull-up enable

0: No pull-up is applied on TAMPALRM output

1: A pull-up is applied on TAMPALRM output

Bit 28 **ALRBFCLR**: Alarm B flag automatic clear

0: Alarm B event generates a trigger event and ALRBF must be cleared by software to allow next alarm event.

1: Alarm B event generates a trigger event. ALRBF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 27 **ALRAFCLR**: Alarm A flag automatic clear

0: Alarm A event generates a trigger event and ALRAF must be cleared by software to allow next alarm event.

1: Alarm A event generates a trigger event. ALRAF is automatically cleared by hardware after 1 ck_apre cycle.
Bit 26 TAMPOE: Tamper detection output enable on TAMPALRM
0: The tamper flag is not routed on TAMPALRM
1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 TAMPTS: Activate timestamp on tamper detection event
0: Tamper detection event does not cause a RTC timestamp to be saved
1: Save RTC timestamp on tamper detection event
TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set up to 3 ck_apre cycles after the tamper flags.

Note: TAMPTS must be cleared before entering RTC initialization mode.

Bit 24 ITSE: timestamp on internal event enable
0: internal event timestamp disabled
1: internal event timestamp enabled

Bit 23 COE: Calibration output enable
This bit enables the CALIB output
0: Calibration output disabled
1: Calibration output enabled

Bits 22:21 OSEL[1:0]: Output selection
These bits are used to select the flag to be routed to TAMPALRM output.
00: Output disabled
01: Alarm A output enabled
10: Alarm B output enabled
11: Wake-up output enabled

Bit 20 POL: Output polarity
This bit is used to configure the polarity of TAMPALRM output.
0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).
1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

Bit 19 COSEL: Calibration output selection
When COE = 1, this bit selects which signal is output on CALIB.
0: Calibration output is 512 Hz
1: Calibration output is 1 Hz
These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to Section 46.3.18: Calibration clock output.

Bit 18 BKP: Backup
This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 SUB1H: Subtract 1 hour (winter time change)
When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.
Setting this bit has no effect when current hour is 0.
0: No effect
1: Subtracts 1 hour to the current time. This can be used for winter time change.
Bit 16  **ADD1H**: Add 1 hour (summer time change)
    When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.
    0: No effect
    1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15  **TSIE**: Timestamp interrupt enable
    0: Timestamp interrupt disable
    1: Timestamp interrupt enable

Bit 14  **WUTIE**: Wake-up timer interrupt enable
    0: Wake-up timer interrupt disabled
    1: Wake-up timer interrupt enabled

Bit 13  **ALRBIE**: Alarm B interrupt enable
    0: Alarm B interrupt disable
    1: Alarm B interrupt enable

Bit 12  **ALRAIE**: Alarm A interrupt enable
    0: Alarm A interrupt disabled
    1: Alarm A interrupt enabled

Bit 11  **TSE**: timestamp enable
    0: timestamp disable
    1: timestamp enable

Bit 10  **WUTE**: Wake-up timer enable
    0: Wake-up timer disabled
    1: Wake-up timer enabled

    *Note*: When the wake-up timer is disabled, wait for WUTWF = 1 before enabling it again.

Bit 9  **ALRB**: Alarm B enable
    0: Alarm B disabled
    1: Alarm B enabled

Bit 8  **ALRA**: Alarm A enable
    0: Alarm A disabled
    1: Alarm A enabled

Bit 7  **SSRUIE**: SSR underflow interrupt enable
    0: SSR underflow interrupt disabled
    1: SSR underflow interrupt enabled

Bit 6  **FMT**: Hour format
    0: 24 hour/day format
    1: AM/PM hour format

Bit 5  **BYP SHAD**: Bypass the shadow registers
    0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLOCK cycles.
    1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.

    *Note*: If the frequency of the APB clock is less than seven times the frequency of RTCCLOCK, BYPSHAD must be set to 1.
Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)
0: RTC_REFIN detection disabled
1: RTC_REFIN detection enabled

*Note*: BIN must be 0x00 and PREDIV_S must be 0x00FF.

Bit 3 **TSEdge**: Timestamp event active edge
0: RTC_TS input rising edge generates a timestamp event
1: RTC_TS input falling edge generates a timestamp event

TSE must be reset when TSEdge is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: CK_wut wake-up clock selection
000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU.
11x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU. Furthermore, $2^{16}$ is added to the WUT counter value.

*Note*: Bits 6 and 4 of this register can be written in initialization mode only (**RTC_ICSR/INITF** = 1).

**WUT** = wake-up unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1] = 11.

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

*It is recommended not to change the hour during the calendar hour increment as it may mask the incrementation of the calendar hour.*

**ADD1H and SUB1H changes are effective in the next second.**

### 46.6.8 RTC privilege mode control register (**RTC_PRIVCFGR**)  
This register can be written only when the APB access is privileged. This register can be write-protected, or each bit of this register can be individually write-protected against nonsecure access depending on the RTC_SECCFGR configuration (refer to **Section 46.3.5: RTC privilege protection modes**).

Address offset: 0x1C
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:16 Reserved, must be kept at reset value.

Bit 15 PRIV: RTC privilege protection
0: All RTC registers can be written when the APB access is privileged or non-privileged, except the registers protected by other privilege protection bits.
1: All RTC registers can be written only when the APB access is privileged.

Bit 14 INITPRIV: Initialization privilege protection
0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is privileged or non-privileged.
1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is privileged.

Bit 13 CALPRIV: Shift register, Delight saving, calibration and reference clock privilege protection
0: Shift register, Delight saving, calibration and reference clock can be written when the APB access is privileged or non-privileged.
1: Shift register, Delight saving, calibration and reference clock can be written only when the APB access is privileged.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 TSPRIV: Timestamp privilege protection
0: RTC Timestamp configuration and interrupt clear can be written when the APB access is privileged or non-privileged.
1: RTC Timestamp configuration and interrupt clear can be written only when the APB access is privileged.

Bit 2 WUTPRIV: Wake-up timer privilege protection
0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is privileged or non-privileged.
1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is privileged.

Bit 1 ALRBPRIV: Alarm B privilege protection
0: RTC Alarm B configuration and interrupt clear can be written when the APB access is privileged or non-privileged.
1: RTC Alarm B configuration and interrupt clear can be written only when the APB access is privileged.

Bit 0 ALRAPRIV: Alarm A and SSR underflow privilege protection
0: RTC Alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is privileged or non-privileged.
1: RTC Alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is privileged.

Note: Refer to Section 46.3.5: RTC privilege protection modes for details on the read protection.
46.6.9 RTC secure configuration register (RTC_SECCFGR)

This register can be written only when the APB access is secure.

This register can be globally write-protected, or each bit of this register can be individually write-protected against non-privileged access depending on the RTC_PRIVCFGR configuration (refer to Section 46.3.5: RTC privilege protection modes).

Address offset: 0x20
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEC</td>
<td>INIT</td>
<td>CAL</td>
<td>SEC</td>
<td>SEC</td>
<td>SEC</td>
<td>SEC</td>
<td>SEC</td>
<td>SEC</td>
<td>SEC</td>
<td>SEC</td>
<td>SEC</td>
<td>SEC</td>
<td>SEC</td>
<td>SEC</td>
<td>SEC</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 SEC: RTC global protection
0: All RTC registers can be written when the APB access is secure or non-secure, except the registers protected by other secure protection bits.
1: All RTC registers can be written only when the APB access is secure.

Bit 14 INITSEC: Initialization protection
0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is secure or nonsecure.
1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is secure.

Bit 13 CALSEC: Shift register, daylight saving, calibration and reference clock protection
0: Shift register, daylight saving, calibration and reference clock can be written when the APB access is secure or nonsecure.
1: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is secure.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 TSSEC: Timestamp protection
0: RTC timestamp configuration and interrupt clear can be written when the APB access is secure or nonsecure.
1: RTC timestamp configuration and interrupt clear can be written only when the APB access is secure.
Bit 2 **WUTSEC**: Wake-up timer protection
0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is secure or nonsecure.
1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is secure.

Bit 1 **ALRBSEC**: Alarm B protection
0: RTC alarm B configuration and interrupt clear can be written when the APB access is secure or nonsecure.
1: RTC alarm B configuration and interrupt clear can be written only when the APB access is secure.

Bit 0 **ALRASEC**: Alarm A and SSR underflow protection
0: RTC alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is secure or nonsecure.
1: RTC alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is secure.

**Note:** Refer to Section 46.3.4: RTC secure protection modes for details on the read protection.

### 46.6.10 RTC write protection register (RTC_WPR)

Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.
Bits 7:0 **KEY[7:0]**: Write protection key
This byte is written by software.
Reading this byte always returns 0x00.
Refer to RTC register write protection for a description of how to unlock RTC register write protection.
46.6.11 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in RTC register write protection on page 1961.

This register can be write-protected against nonsecure access. Refer to Section 46.3.4: RTC secure protection modes.

This register can be write-protected against non-privileged access. Refer to Section 46.3.5: RTC privilege protection modes.

Address offset: 0x28
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CALP</td>
<td>CALW8</td>
<td>CALW16</td>
<td>LPCAL</td>
<td>Res.</td>
<td>Res.</td>
<td>Res.</td>
<td>CALM[8:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm
0: No RTCCCLK pulses are added.
1: One RTCCCLK pulse is effectively inserted every $2^{41}$ pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. If the input frequency is 32768 Hz, the number of RTCCCLK pulses added during a 32-second window is calculated as follows:

$$(512 \times \text{CALP}) - \text{CALM}.$$ 

Refer to Section 46.3.16: RTC smooth digital calibration.

Bit 14 **CALW8**: Use an 8-second calibration cycle period
When CALW8 is set to 1, the 8-second calibration cycle period is selected.

*Note*: **CALM[1:0]** are stuck at 00 when CALW8 = 1. Refer to Section 46.3.16: RTC smooth digital calibration.

Bit 13 **CALW16**: Use a 16-second calibration cycle period
When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.

*Note*: **CALM[0]** is stuck at 0 when CALW16 = 1. Refer to Section 46.3.16: RTC smooth digital calibration.
Bit 12  **LPCAL**: RTC low-power mode  
  0: Calibration window is 2<sup>20</sup> RTCCLK, which is a high-consumption mode. This mode must be set only when less than 32s calibration window is required.  
  1: Calibration window is 2<sup>20</sup> ck_apre, which is the required configuration for ultra-low consumption mode.  

Bits 11:9  Reserved, must be kept at reset value.  

Bits 8:0  **CALM[8:0]**: Calibration minus  
The frequency of the calendar is reduced by masking CALM out of 2<sup>20</sup> RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.  
To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See [Section 46.3.16: RTC smooth digital calibration on page 1966](#).  

### 46.6.12 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 1961](#).  

This register can be protected against nonsecure access. Refer to [Section 46.3.4: RTC secure protection modes](#).  

This register can be protected against non-privileged access. Refer to [Section 46.3.5: RTC privilege protection modes](#).  

Address offset: 0x2C  
Backup domain reset value: 0x0000 0000  
System reset: not affected  

<table>
<thead>
<tr>
<th>Address (hex)</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADD1S</strong></td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>SUBFS[14:0]</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

---

**Note**: This is a partial representation of the document. The full content includes additional technical details and protection modes, which are not fully captured here due to the nature of the extract provided.
46.6.13 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against nonsecure access. Refer to Section 46.3.4: RTC secure protection modes.

This register can be protected against non-privileged access. Refer to Section 46.3.5: RTC privilege protection modes.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  |

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 PM: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 HT[1:0]: Hour tens in BCD format.

Bits 19:16 HU[3:0]: Hour units in BCD format.
### 46.6.14 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against nonsecure access. Refer to Section 46.3.4: RTC secure protection modes.

This register can be protected against non-privileged access. Refer to Section 46.3.5: RTC privilege protection modes.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:16  Reserved, must be kept at reset value.

Bits 15:13  **WDU[2:0]**: Week day units

Bit 12  **MT**: Month tens in BCD format

Bits 11:8  **MU[3:0]**: Month units in BCD format

Bits 7:6  Reserved, must be kept at reset value.

Bits 5:4  **DT[1:0]**: Date tens in BCD format

Bits 3:0  **DU[3:0]**: Date units in BCD format
46.6.15 **RTC timestamp subsecond register (RTC_TSSSR)**

The content of this register is valid only when TSF is set to 1 in RTC SR. It is cleared when the TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x38
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>r</td>
</tr>
<tr>
<td>30</td>
<td>r</td>
</tr>
<tr>
<td>29</td>
<td>r</td>
</tr>
<tr>
<td>28</td>
<td>r</td>
</tr>
<tr>
<td>27</td>
<td>r</td>
</tr>
<tr>
<td>26</td>
<td>r</td>
</tr>
<tr>
<td>25</td>
<td>r</td>
</tr>
<tr>
<td>24</td>
<td>r</td>
</tr>
<tr>
<td>23</td>
<td>r</td>
</tr>
<tr>
<td>22</td>
<td>r</td>
</tr>
<tr>
<td>21</td>
<td>r</td>
</tr>
<tr>
<td>20</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>r</td>
</tr>
<tr>
<td>18</td>
<td>r</td>
</tr>
<tr>
<td>17</td>
<td>r</td>
</tr>
<tr>
<td>16</td>
<td>r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0: SS[31:0]: Subsecond value/synchronous binary counter values
SS[31:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

46.6.16 **RTC alarm A register (RTC_ALMAR)**

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x40
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bit 31  **MSK4**: Alarm A date mask  
0: Alarm A set if the date/day match  
1: Date/day don’t care in alarm A comparison  

Bit 30  **WDSEL**: Week day selection  
0: DU[3:0] represents the date units  
1: DU[3:0] represents the week day. DT[1:0] is don’t care.  

Bits 29:28  **DT[1:0]**: Date tens in BCD format  

Bits 27:24  **DU[3:0]**: Date units or day in BCD format  

Bit 23  **MSK3**: Alarm A hours mask  
0: Alarm A set if the hours match  
1: Hours don’t care in alarm A comparison  

Bit 22  **PM**: AM/PM notation  
0: AM or 24-hour format  
1: PM  

Bits 21:20  **HT[1:0]**: Hour tens in BCD format  

Bits 19:16  **HU[3:0]**: Hour units in BCD format  

Bit 15  **MSK2**: Alarm A minutes mask  
0: Alarm A set if the minutes match  
1: Minutes don’t care in alarm A comparison  

Bits 14:12  **MNT[2:0]**: Minute tens in BCD format  

Bits 11:8  **MNU[3:0]**: Minute units in BCD format  

Bit 7  **MSK1**: Alarm A seconds mask  
0: Alarm A set if the seconds match  
1: Seconds don’t care in alarm A comparison  

Bits 6:4  **ST[2:0]**: Second tens in BCD format.  

Bits 3:0  **SU[3:0]**: Second units in BCD format.
46.6.17  **RTC alarm A subsecond register (RTC_ALRM ASSR)**

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x44

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSCLR</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>MASKSS[5:0]</strong></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

- **Bit 31** **SSCLR**: Clear synchronous counter on alarm (Binary mode only)
  - 0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.
  - 1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRABINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one clock cycle after reaching RTC_ALRABINR.SS[31:0].

  *Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).*

- **Bit 30** Reserved, must be kept at reset value.

- **Bits 29:24** **MASKSS[5:0]**: Mask the most-significant bits starting at this bit
  - 0: No comparison on subseconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
  - 1: SS[31:1] are don’t care in Alarm A comparison. Only SS[0] is compared.
  - ...
  - 31: SS[31] is don’t care in Alarm A comparison. Only SS[30:0] are compared.

  *Note: In BCD mode (BIN=00) the overflow bits of the synchronous counter (bits 31:15) are never compared. These bits can be different from 0 only after a shift operation.*

- **Bits 23:15** Reserved, must be kept at reset value.

- **Bits 14:0** **SS[14:0]**: Subseconds value

  This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

  This field is the mirror of SS[14:0] in the RTC_ALRABINR, and so can also be read or written through RTC_ALRABINR.

  *Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAM ATCR1.*
46.6.18  RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to Section 46.3.4: RTC secure protection modes.

This register can be protected against non-privileged access. Refer to Section 46.3.5: RTC privilege protection modes.

Address offset: 0x48
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

- **Bit 31** MSK4: Alarm B date mask
  - 0: Alarm B set if the date and day match
  - 1: Date and day don’t care in alarm B comparison

- **Bit 30** WDSEL: Week day selection
  - 0: DU[3:0] represents the date units
  - 1: DU[3:0] represents the week day. DT[1:0] is don’t care.

- **Bits 29:28** DT[1:0]: Date tens in BCD format

- **Bits 27:24** DU[3:0]: Date units or day in BCD format

- **Bit 23** MSK3: Alarm B hours mask
  - 0: Alarm B set if the hours match
  - 1: Hours don’t care in alarm B comparison

- **Bit 22** PM: AM/PM notation
  - 0: AM or 24-hour format
  - 1: PM

- **Bits 21:20** HT[1:0]: Hour tens in BCD format

- **Bits 19:16** HU[3:0]: Hour units in BCD format

- **Bit 15** MSK2: Alarm B minutes mask
  - 0: Alarm B set if the minutes match
  - 1: Minutes don’t care in alarm B comparison

- **Bits 14:12** MNT[2:0]: Minute tens in BCD format

- **Bits 11:8** MNU[3:0]: Minute units in BCD format
Bit 7  **MSK1**: Alarm B seconds mask  
0: Alarm B set if the seconds match  
1: Seconds don’t care in alarm B comparison  

Bits 6:4  **ST[2:0]**: Second tens in BCD format  

Bits 3:0  **SU[3:0]**: Second units in BCD format  

### 46.6.19  RTC alarm B subsecond register (RTC_ALRMBSSR)  

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.  

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.  

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.  

Address offset: 0x4C  
Backup domain reset value: 0x0000 0000  
System reset: not affected  

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**SSCLR**: Clear synchronous counter on alarm (Binary mode only)  
0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.  
1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRBBINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one clock_cycle after reaching RTC_ALRBBINR.SS[31:0].  

*Note:* **SSCLR** must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).  

Bit 30  Reserved, must be kept at reset value.
Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
1: SS[31:1] are don’t care in Alarm B comparison. Only SS[0] is compared.
2: SS[31:2] are don’t care in Alarm B comparison. Only SS[1:0] are compared.
...
31: SS[31] is don’t care in Alarm B comparison. Only SS[30:0] are compared.

*Note:* In BCD mode (BIN=00) the overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRBBINR, and so can also be read or written through RTC_ALRBBINR.


### 46.6.20 RTC status register (RTC_SR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x50

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SSR UF</td>
<td>ITSF</td>
<td>TSOVF</td>
<td>TSF</td>
<td>WUTF</td>
<td>ALRB</td>
<td>ALRAF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

**Bit 6 SSRUF:** SSR underflow flag

This flag is set by hardware when the SSR rolls under 0. SSRUF is not set when SSCLR=1.

**Bit 5 ITSF:** Internal timestamp flag

This flag is set by hardware when a timestamp on the internal event occurs.
Real-time clock (RTC)  

<table>
<thead>
<tr>
<th>Bit 4</th>
<th>TSOVF: Timestamp overflow flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This flag is set by hardware when a timestamp event occurs while TSF is already set.</td>
</tr>
<tr>
<td></td>
<td>It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>TSF: Timestamp flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This flag is set by hardware when a timestamp event occurs.</td>
</tr>
<tr>
<td></td>
<td>If ITSF flag is set, TSF must be cleared together with ITSF.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> TSF is not set if TAMPTS = 1 and the tamper flag is read during the 3 ck_apre cycles following tamper event. Refer to Timestamp on tamper event for more details.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>WUTF: Wake-up timer flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This flag is set by hardware when the wake-up auto-reload counter reaches 0.</td>
</tr>
<tr>
<td></td>
<td>If WUTOCLR[15:0] is different from 0x0000, WUTF is cleared by hardware when the wake-up auto-reload counter reaches WUTOCLR value.</td>
</tr>
<tr>
<td></td>
<td>If WUTOCLR[15:0] is 0x0000, WUTF must be cleared by software.</td>
</tr>
<tr>
<td></td>
<td>This flag must be cleared by software at least 1.5 RTCCLOCK periods before WUTF is set to 1 again.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>ALRBF: Alarm B flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>ALRAF: Alarm A flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).</td>
</tr>
</tbody>
</table>

**Note:** The bits of this register are cleared 2 APB clock cycles after setting their corresponding clear bit in the RTC_SCR register.

### 46.6.21 RTC nonsecure masked interrupt status register (RTC_MISR)

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to Section 46.3.5: RTC privilege protection modes.

Address offset: 0x54
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Address</th>
<th>SSR</th>
<th>UMF</th>
<th>ITS</th>
<th>MF</th>
<th>TSOVF</th>
<th>MF</th>
<th>TS</th>
<th>MF</th>
<th>WUTF</th>
<th>MF</th>
<th>ALRB</th>
<th>MF</th>
<th>ALRA</th>
<th>MF</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23-16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15-8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7-0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:7  Reserved, must be kept at reset value.

Bit 6  **SSRUMF**: SSR underflow nonsecure masked flag
This flag is set by hardware when the SSR underflow nonsecure interrupt occurs.

Bit 5  **ITSMF**: Internal timestamp nonsecure masked flag
This flag is set by hardware when a timestamp on the internal event occurs and timestamp nonsecure interrupt is raised.

Bit 4  **TSOVMF**: Timestamp overflow nonsecure masked flag
This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3  **TSMF**: Timestamp nonsecure masked flag
This flag is set by hardware when a timestamp nonsecure interrupt occurs. If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2  **WUTMF**: Wake-up timer nonsecure masked flag
This flag is set by hardware when the wake-up timer nonsecure interrupt occurs. This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1  **ALRBMF**: Alarm B nonsecure masked flag
This flag is set by hardware when the alarm B nonsecure interrupt occurs.

Bit 0  **ALRAMF**: Alarm A masked flag
This flag is set by hardware when the alarm A nonsecure interrupt occurs.

### 46.6.22 RTC secure masked interrupt status register (RTC_SMISR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to **Section 46.3.4: RTC secure protection modes**.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to **Section 46.3.5: RTC privilege protection modes**.

Address offset: 0x58
Backup domain reset value: 0x0000 0000
System reset: not affected
Bits 31:7  Reserved, must be kept at reset value.

Bit 6  **SSRUMF**: SSR underflow secure masked flag
This flag is set by hardware when the SSR underflow secure interrupt occurs.

Bit 5  **ITSMF**: Internal timestamp interrupt secure masked flag
This flag is set by hardware when a timestamp on the internal event occurs and timestamp secure interrupt is raised.

Bit 4  **TSOVMF**: Timestamp overflow interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3  **TSMF**: Timestamp interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2  **WUTMF**: Wake-up timer interrupt secure masked flag
This flag is set by hardware when the wake-up timer secure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1  **ALRBMF**: Alarm B interrupt secure masked flag
This flag is set by hardware when the alarm B secure interrupt occurs.

Bit 0  **ALRAMF**: Alarm A interrupt secure masked flag
This flag is set by hardware when the alarm A secure interrupt occurs.

### 46.6.23  RTC status clear register (RTC_SCR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to Section 46.3.4: RTC secure protection modes.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to Section 46.3.5: RTC privilege protection modes.

Address offset: 0x5C
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>
46.6.24 RTC option register (RTC_OR)

Address offset: 0x60

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **OUT2_RMP**: RTC_OUT2 mapping
0: RTC_OUT2 is mapped on PI8
1: RTC_OUT2 is mapped on PB2
### 46.6.25 RTC alarm A binary mode register (RTC_ALRABINR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to Section 46.3.4: RTC secure protection modes.

This register can be protected against non-privileged access. Refer to Section 46.3.5: RTC privilege protection modes.

Address offset: 0x70

Backup domain reset value: 0x0000 0000

System reset: not affected

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**SS[31:16]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMASSRR, and so can also be read or written through RTC_ALRMASSR.


### 46.6.26 RTC alarm B binary mode register (RTC_ALRBBINR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to Section 46.3.4: RTC secure protection modes.

This register can be protected against non-privileged access. Refer to Section 46.3.5: RTC privilege protection modes.

Address offset: 0x74

Backup domain reset value: 0x0000 0000

System reset: not affected

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

---
Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm Bis to be activated. Only bits 0 up MASKSS-1 are compared. SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMBSSRR, and so can also be read or written through RTC_ALRMBSSR.

**Note:** SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.
### 46.6.27 RTC register map

#### Table 486. RTC register map and reset values

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>RTC_TR</td>
<td>PM</td>
<td>HT [1:0]</td>
<td>HU[3:0]</td>
<td>MNT[2:0]</td>
<td>MNU[3:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x04</td>
<td>RTC_DR</td>
<td>YT[3:0]</td>
<td>YU[3:0]</td>
<td>WDU[2:0]</td>
<td>MT</td>
<td>MU[3:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x08</td>
<td>RTC_SSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0C</td>
<td>RTC_ICSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x10</td>
<td>RTC_PRER</td>
<td>PREDIV_A[6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0x14</td>
<td>RTC_WUTR</td>
<td>WUTOCLR[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x18</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x1C</td>
<td>RTC_PRIVCFGR</td>
<td>PRIV</td>
<td>INITPRIV</td>
<td>CALPRIV</td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x20</td>
<td>RTC_SECCFGR</td>
<td>SEC</td>
<td>INITSEC</td>
<td>CALSEC</td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x24</td>
<td>RTC_WPR</td>
<td>KEY[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x28</td>
<td>RTC_CALR</td>
<td>CALP</td>
<td>CALW8</td>
<td>CALW16</td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x2C</td>
<td>RTC_SHIFTR</td>
<td>AD1S</td>
<td>SUBFS[14:0]</td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>
Table 486. RTC register map and reset values (continued)

| Offset | Register name     | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x30   | RTC_TSTR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x34   | RTC_TSDR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x38   | RTC_TSSSR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x40   | RTC_ALRMAR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x44   | RTC_ALRMASSR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x48   | RTC_ALRMBSR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x4C   | RTC_ALRMBSSR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x50   | RTC_SR           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x54   | RTC_MISR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x58   | RTC_SMISR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x5C   | RTC_SCR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x60   | RTC_OR           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x70   | RTC_ALRABINR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |

Table 486. RTC register map and reset values (continued)
Refer to \textit{Section 2.3} for the register boundary addresses.
47 Tamper and backup registers (TAMP)

47.1 Introduction

The anti-tamper detection circuit is used to protect sensitive data from external attacks. 32 32-bit backup registers are retained in all low-power modes and also in $V_{BAT}$ mode. The backup registers, as well as other secrets in the device, are protected by this anti-tamper detection circuit with 11 tamper pins and 13 internal tampers. The external tamper pins can be configured for edge detection, or level detection with or without filtering, or active tamper which increases the security level by auto checking that the tamper pins are not externally opened or shorted.
47.2 **TAMP main features**

- A tamper detection can optionally erase the backup registers, backup SRAM, SRAM2, caches and cryptographic peripherals. The device resources protected by tamper are named “device secrets”.
- 32 32-bit backup registers:
  - The backup registers (TAMP\_BKP\_xR) are implemented in the backup domain that remains powered-on by \( V_{BAT} \) when the \( V_{DD} \) power is switched off.
- Up to 11 tamper pins for 8 external tamper detection events:
  - Active tamper mode: continuous comparison between tamper output and input to protect from physical open-short attacks.
  - Flexible active tamper I/O management: from 5 meshes (each input associated to its own exclusive output) to 8 meshes (single output shared for up to 8 tamper inputs)
  - Passive tampers: ultra-low power edge or level detection with internal pull-up hardware management.
  - Configurable digital filter.
- 13 internal tamper events to protect against transient or environmental perturbation attacks
- Each tamper can be configured in two modes:
  - Confirmed mode: immediate erase of secrets on tamper detection, including backup registers erase
  - Potential mode: most of the secrets erase following a tamper detection are launched by software
- Any tamper detection can generate a RTC timestamp event.
- TrustZone support:
  - Tamper secure or nonsecure configuration.
  - Backup registers configuration in 3 configurable-size areas:
    - 1 read/write secure area.
    - 1 write secure/read nonsecure area.
    - 1 read/write nonsecure area.
  - Boot hardware key for secure AES, stored in backup registers, protected against read and write access.
- Tamper configuration and backup registers privilege protection
- Monotonic counter.
47.3 TAMP functional description

47.3.1 TAMP block diagram

1. The number of external and internal tampers depends on products.
47.3.2 TAMP pins and internal signals

Table 487. TAMP input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMP_INx (x = pin index)</td>
<td>Input</td>
<td>Tamper input pin</td>
</tr>
<tr>
<td>TAMP_OUTx (x = pin index)</td>
<td>Output</td>
<td>Tamper output pin (active mode only)</td>
</tr>
</tbody>
</table>

Table 488. TAMP internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tamp_ker_ck</td>
<td>Input</td>
<td>TAMP kernel clock, connected to rtc_ker_ck and also named RTCCLOCK in this document</td>
</tr>
<tr>
<td>tamp_pclk</td>
<td>Input</td>
<td>TAMP APB clock, connected to rtc_pclk</td>
</tr>
<tr>
<td>tamp_itamp[y]</td>
<td>Inputs</td>
<td>Internal tamper event sources</td>
</tr>
<tr>
<td>tamp_tzen</td>
<td>Input</td>
<td>TAMP TrustZone enabled</td>
</tr>
<tr>
<td>tamp_evt</td>
<td>Output</td>
<td>Tamper event detection flag (internal or external tamper), whatever confirmed or potential mode configuration.</td>
</tr>
</tbody>
</table>
| tamp_potential       | Output      | Potential tamper detection signal, used for device secrets\(^1\) protection. This signal is active when:  
                     |             | – a tamper event detection flag (internal or external tamper), is generated in potential mode.  
                     |             | – or a software request is done by writing BKBLOCK to 1                     |
| tamp_confirmed       | Output      | Confirmed tamper detection signal, used for device secrets\(^1\) protection. This signal is active when:  
                     |             | – a tamper event detection flag (internal or external tamper), is generated in confirmed mode.  
                     |             | – or a software request is done by writing BKERASE to 1                     |
| tamp_potential_rpcfgz| Output      | Potential tamper detection signal generated only when RPCFGz = 1. This signal is active when:  
                     |             | – a tamper event detection flag (internal or external tamper), is generated in potential mode.  
                     |             | – or a software request is done by writing BKBLOCK to 1                     |
The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some detections modes are not available in some low-power modes or \( V_{BAT} \) depending on the selected clock (refer to Section 47.4: TAMP low-power modes for more details).

### Table 488. TAMP internal input/output signals (continued)

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tamp_confirmed_rpcfgz (z = signal index)</td>
<td>Output</td>
<td>Confirmed tamper detection signal generated only when RPCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in confirmed mode. – or a software request is done by writing BKERASE to 1</td>
</tr>
<tr>
<td>tamp_it</td>
<td>Output</td>
<td>TAMP interrupt (refer to Section 47.5: TAMP interrupts for details)</td>
</tr>
<tr>
<td>tamp_trg[x] (x = signal index)</td>
<td>Output</td>
<td>Tamper detection trigger</td>
</tr>
<tr>
<td>tamp_bhk</td>
<td>Output</td>
<td>Tamper boot hardware key bus</td>
</tr>
</tbody>
</table>

1. Refer to Table 489: TAMP interconnection.

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some detections modes are not available in some low-power modes or \( V_{BAT} \) depending on the selected clock (refer to Section 47.4: TAMP low-power modes for more details).

### Table 489. TAMP interconnection

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>tamp_tzen</td>
<td>From FLASH option bytes: TZEN</td>
</tr>
<tr>
<td>tamp_evt</td>
<td>rtc_tamp_evt used to generate a timestamp event</td>
</tr>
<tr>
<td>tamp_potential</td>
<td>The tamp_potential signal is used to block the read and write accesses to the device secrets listed hereafter: – backup registers – SRAM2 RHUK (root hardware unique key) in system Flash memory and BHK (boot hardware key) hardware buses to SAES are blocked. The tamp_potential signal is used to erase the device secrets listed hereafter: – iCACHE content – SAES, AES, HASH peripherals – PKA SRAM The device secrets access is blocked when erase is ongoing.</td>
</tr>
</tbody>
</table>
The TZEN option bit is used to activate TrustZone in the device.

Table 489. TAMP interconnection (continued)

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tamp_confirmed</code></td>
<td>The <code>tamp_confirmed</code> signal is used to erase the device secrets listed hereafter:</td>
</tr>
<tr>
<td></td>
<td>– backup registers</td>
</tr>
<tr>
<td></td>
<td>– SRAM2</td>
</tr>
<tr>
<td></td>
<td>– ICACHE/DCACHE content</td>
</tr>
<tr>
<td></td>
<td>– OTFDEC keys and CRC registers</td>
</tr>
<tr>
<td></td>
<td>– SAES, AES, HASH peripherals</td>
</tr>
<tr>
<td></td>
<td>– PKA SRAM</td>
</tr>
<tr>
<td></td>
<td>The device secrets access is blocked when erase is ongoing.</td>
</tr>
<tr>
<td></td>
<td>RHUK in system Flash memory (root hardware unique key) hardware bus to SAES is</td>
</tr>
<tr>
<td></td>
<td>blocked.</td>
</tr>
<tr>
<td><code>tamp_potential_rpcfg0</code></td>
<td>When the bit RPCFG0 is set in the TAMP_RPCFGR, the</td>
</tr>
<tr>
<td></td>
<td><code>tamp_potential_rpcfg0</code> signal is used to block the read and write accesses</td>
</tr>
<tr>
<td></td>
<td>to the device secrets listed hereafter:</td>
</tr>
<tr>
<td></td>
<td>– Backup SRAM</td>
</tr>
<tr>
<td><code>tamp_confirmed_rpcfg0</code></td>
<td>When the bit RPCFG0 is set in the TAMP_RPCFGR, the</td>
</tr>
<tr>
<td></td>
<td><code>tamp_confirmed_rpcfg0</code> signal is used to erase the device secrets listed</td>
</tr>
<tr>
<td></td>
<td>hereafter:</td>
</tr>
<tr>
<td></td>
<td>– Backup SRAM</td>
</tr>
<tr>
<td></td>
<td>The device secrets access is blocked when erase is on-going.</td>
</tr>
<tr>
<td><code>tamp_itamp1</code></td>
<td>Backup domain voltage threshold monitoring¹(1)</td>
</tr>
<tr>
<td><code>tamp_itamp2</code></td>
<td>Temperature monitoring¹(1)</td>
</tr>
<tr>
<td><code>tamp_itamp3</code></td>
<td>LSE monitoring (LSECSS)²(2)</td>
</tr>
<tr>
<td><code>tamp_itamp4</code></td>
<td>HSE monitoring (rcc_hsecss_fail)</td>
</tr>
<tr>
<td><code>tamp_itamp5</code></td>
<td>RTC calendar overflow (rtc_calovf)</td>
</tr>
<tr>
<td><code>tamp_itamp6</code></td>
<td>JTAG/SWD access when NVSTATE ≠ OPEN</td>
</tr>
<tr>
<td><code>tamp_itamp7</code></td>
<td>ADC2 watchdog monitoring 1</td>
</tr>
<tr>
<td><code>tamp_itamp8</code></td>
<td>Monotonic counter 1 overflow</td>
</tr>
<tr>
<td><code>tamp_itamp9</code></td>
<td>Cryptographic peripherals fault (SAES or AES or PKA or TRNG)</td>
</tr>
<tr>
<td><code>tamp_itamp11</code></td>
<td>IWDG reset when tamper flag is set (potential tamper timeout)</td>
</tr>
<tr>
<td><code>tamp_itamp12</code></td>
<td>ADC2 analog watchdog monitoring 2</td>
</tr>
<tr>
<td><code>tamp_itamp13</code></td>
<td>ADC2 analog watchdog monitoring 3</td>
</tr>
<tr>
<td><code>tamp_itamp15</code></td>
<td>System fault</td>
</tr>
<tr>
<td><code>tamp_bhk</code></td>
<td>saes_bhk. This bus is used to load the boot hardware key in the secure AES</td>
</tr>
<tr>
<td></td>
<td>co-processor.</td>
</tr>
</tbody>
</table>

1. This monitoring must be enabled by setting MONEN in `PWR backup domain control register (PWR_BDCR)`.  
2. This monitoring must be enabled by setting LSEON in `PWR backup domain control register (PWR_BDCR)`.  
3. This signal is generated in the TAMP peripheral.

The TZEN option bit is used to activate TrustZone in the device.
TZEN = 1: TrustZone activated.
TZEN = 0: TrustZone disabled.

When TrustZone is disabled, the APB access to the TAMP registers are nonsecure.

### 47.3.3 GPIOs controlled by the RTC and TAMP

Refer to Section 46.3.3: GPIOs controlled by the RTC and TAMP.

### 47.3.4 TAMP register write protection

After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable TAMP registers write access.

### 47.3.5 TAMP secure protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both secure and nonsecure modes, except for the TAMP secure configuration register (TAMP_SECCFGR) which can be written in secure mode only. The TAMP protection configuration is not affected by a system reset.

- When the TAMPSEC bit is set in the TAMP_SECCFGR register:
  - Writing the TAMP registers is possible only in secure mode, except for the backup registers which have their own protection setting.
  - Reading TAMP_SECCFGR, TAMP_PRIVCFGR and TAMP_MISR is always possible in secure and nonsecure modes. All the other TAMP registers can be read only in secure mode, except for the backup registers and monotonic counters which have their own protection setting.

- When the CNT1SEC bit is set in the TAMP_SECCFGR register: the TAMP_COUNT1R can be read and written only in secure mode.

A nonsecure access to a secure-protected register is denied:

- There is no bus error generated.
- A notification is generated through a flag/interrupt in the TZIC (TrustZone illegal access controller).
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the TAMP reset and clock control is also secured in the RCC.
47.3.6 Backup registers protection zones

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPWSEC[7:0] (refer to the figure below):

![Figure 619. Backup registers protection zones](image)

1. \( l \) = last backup register index

In case TZEN =1, the bits BKPWPRIV and BKPRWPRIV in the TAMP_PRIVCFGR can be written only in secure mode.

47.3.7 TAMP privilege protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both privileged and non-privileged modes, except for the TAMP privilege configuration register (TAMP_PRIVCFGR) which can be written in privilege mode only. The TAMP protection configuration is not affected by a system reset.

When the TAMPPRIV bit is set in the TAMP_PRIVCFGR register:
- Writing the TAMP registers is possible only in privilege mode, except for the backup registers and the monotonic counters which have their own protection setting.
- When the CNT1PRIV bit is set in the TAMP_PRIVCFGR register: the TAMP_COUNT1R can be read and written only in privilege mode.
- Reading TAMP_SECCFGR, TAMP_PRIVCFGR is always possible in privilege and non-privileged modes. All the other TAMP registers can be read only in privileged mode, except for the backup registers and the monotonic counters which have their own protection setting.

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPRWPRIV for the protection zone 1, and thanks to BKPRWSEC[7:0], BKPWSEC[7:0] and BKPRWPRIV for the protection zone 2 (refer to Figure 619). The BHKLOCK bit can be written only in privileged mode when the BKPRWPRIV bit is set.

A non-privileged access to a privileged-protected register is denied:
- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.
47.3.8 Boot hardware key (BHK)

The eight first backup registers from TAMP_BKP0R to TAMP_BKP7R can be used to store a boot hardware key for the secure AES.

For this purpose, these registers must belong to the Protection Zone 1: BKPRWSEC must be greater or equal to 8.

Once the backup registers are written with the boot hardware key, the BHKLOCK bit must be set in the TAMP_SECCFGR register. Once BHKLOCK is set, the 8 backup registers cannot be accessed anymore by software: they are read as 0 and write to these registers is ignored. BHKLOCK cannot be cleared by software, and is cleared by hardware following a tamper event or when the readout protection (RDP) is disabled. It is also cleared with BKERASE command (in all cases the backup registers are also erased).

Refer to section secure AES co-processor (SAES) for details on procedure to download the boot hardware key in the SAES.

47.3.9 Tamper detection

The tamper detection main purpose is to protect the device secrets from device external attacks. The detection is made on events on TAMP_INx (x = pin index) I/Os, or on internal monitors detecting out-of-range device conditions.

The tamper detection can be configured for the following purposes:
- erase the backup registers and other device secrets stored in SRAMs or peripherals listed in Table 489: TAMP interconnection. The device secrets list is configurable thanks to TAMP resources protection configuration register (TAMP_RPCFGR).
- block the read/write access to the backup registers and other device secrets stored in SRAMs or peripherals listed in Table 489: TAMP interconnection. The device secrets list is configurable thanks to TAMP_RPCFGR.
- generate an interrupt, capable to wake-up from low-power modes
- generate a hardware trigger for the low-power timers, or a RTC timestamp event

The external I/Os tamper detection supports 2 main configurations:
- Passive mode: TAMP_INx I/Os are monitored and a tamper is detected either on edge or on level.
- Active mode: TAMP_INx (x = pin index) is continuously compared with TAMP_OUTy (y = pin index) allowing open-short detection.

A digital filter can be applied on external tamper detection to avoid false detection. In addition, it is possible to configure each tamper source in potential mode, so that the secrets erase is not launched by hardware on tamper detection. The secrets erase can then be launched by software after software checks.

47.3.10 TAMP backup registers and other device secrets erase

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers and the other device secrets are not reset when the corresponding mask is set (TAMPxMSK=1 in the TAMP_CR2 register).

Note: The backup registers are also erased when the readout protection of the flash is changed from level 1 to level 0.
Tamper and backup registers (TAMP)

Tamper detection – confirmed mode

The confirmed mode is selected for TAMPx (external tamper x) when TAMPxPOM = 0 in the TAMP_CR2 register. The confirmed mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 0 in the TAMP_CR3 register. The effects of a tamper detection in confirmed mode are described with tamp_confirmed and tamp_confirmed_rpcfgx signals in the Table 489: TAMP interconnection.

This mode is selected to erase automatically the device secrets when the tamper is detected.

Tamper detection – potential mode

The potential mode is selected for TAMPx (external tamper x) when TAMPxPOM = 1 in the TAMP_CR2 register. The potential tamper mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 1 in the TAMP_CR3 register. The effects of a tamper detection in potential mode are described with tamp_potential and tamp_potential_rpcfgx signals in the Table 489: TAMP interconnection.

This mode is selected to avoid irreversible erasure of some device secrets when the tamper is detected. In this mode, some device secrets are not erased when the corresponding tamper event is detected. In addition, the read and write accesses to these device secrets are blocked as soon as the tamper detection flag is set in potential mode, until this flag is cleared by setting the corresponding clear flag in the TAMP_SCR register. Therefore the software can perform some checks to discriminate false from true tampers, and decide to launch secrets erase only in case of the potential tamper is confirmed to be a true tamper. The device secrets are erased by software by setting the BKERASE bit in the TAMP_CR2 register.

Potential tamper to confirmed tamper timeout

Some internal tampers generate a tamper event if the independent watchdog reset occurs when another tamper flag is set (refer to Table 489: TAMP interconnection). The IWDG tamper must be configured with ITAMPxPOM = 0. This permits the erasure of device secrets to be forced by hardware after a timeout, in case the previous tamper event was in potential mode. This is equivalent to change the “potential tamper” into “confirmed tamper” if a watchdog reset occurs before any software decision following the potential tamper event.

Device resources protection configuration

Some device resources can be configured in order to be included to the list of the device secrets protected by tamper detection.

When RPCFGz = 0 in the TAMP_RPCFGR, the device resource associated to RPCFGz is not protected by the TAMP peripheral:
- It is not affected by tamper detection (whatever confirmed or potential mode)
- It is not affected by BKERASE software command
- It is not affected by BKBLOCK software command

When RPCFGz = 1 in the TAMP_RPCFGR, the device resource associated to RPCFGz is protected by the TAMP peripheral:
- It is affected by confirmed tamper detection and BKERASE software command, as described with tamp_confirmed_rpcfgz signal in Table 489: TAMP interconnection
- It is affected by potential tamper detection and BKBLOCK software command, as described with tamp_potential_rpcfgz signal in Table 489: TAMP interconnection
Device secrets access blocked by software

By default, the device secrets can be accessed by the application, except if a tamper event flag is detected: the device secrets access is not possible as long as a tamper flag is set.

It is possible to block the access to the device secrets by software, by setting the BKBLOCK bit of the TAMP_CR2 register. The device secrets access is possible only when BKBLOCK = 0 and no tamper flag is set.

### 47.3.11 Tamper detection configuration and initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP_CR register.

Each TAMP_INx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

By setting the TAMPxIE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

**Trigger output generation on tamper event**

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit is cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration permits the low-power timers to be triggered automatically in Stop mode, without requiring the system wake-up to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in the Level detection with filtering on tamper inputs (passive mode) mode (TAMPFLT ≠ 00 and active mode is not selected).

**Timestamp on tamper event**

With TAMPPTS set to 1 in the RTC_CR, any internal or external tamper event causes a timestamp to occur. In case a timestamp occurs due to tamper event, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a normal timestamp event occurs.

**Note:** TSF is set up to 3 ck_apre cycles after TAMPxF flags. TSF is not set if RTCCCLK is stopped (it is set when RTCCCLK restarts).
Note: If TAMPxF is cleared before the expected rise of TSF, TSF is not set. Consequently, in case TAMPTS = 1, the software should either wait for timestamp flag before clearing the tamper flag, or should read the RTC counters values in the TAMP interrupt routine.

Edge detection on tamper inputs (passive mode)

If the TAMPFILT bits are 00, the TAMPI_nX pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMPI_nX inputs are deactivated when edge detection is selected.

Caution: When TAMPFILT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMPI_nX should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (TAMP_BKPnX). This prevents the application from writing to the backup registers while the TAMPI_nX input value still indicates a tamper detection. This is equivalent to a level detection on the TAMPI_nX input.

Note: Tamper detection is still active when V_DD power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPx is mapped should be externally tied to the correct level.

Level detection with filtering on tamper inputs (passive mode)

Level detection with filtering is performed by setting TAMPFILT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFILT) consecutive samples are observed at the level designated by the TAMPI_nX bits.

The TAMPI_nX inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMPI_nX inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

Note: Refer to the microcontroller datasheet for the electrical characteristics of the pull-up resistors.

Active tamper detection

When the TAMPxAM bit is set in the TAMP_ATCR, the tamper events are configured in active mode, which is based on a comparison between a TAMP_OUTy pin and a TAMPI_nX pin. By default (ATOSHARE = 0) the comparison is made between TAMPI_nX and TAMP_OUTx (y = x). When ATOSHARE bit is set, the same output can be used for several tamper inputs. The TAMP_OUTy function is enabled on the I/O as soon as it is selected for comparison with an active tamper input TAMPI_nX (TAMPxEN = TAMPxAM = 1), thanks to ATOSHARE and ATOSELx bits. Refer to ATOSHARE and ATOSELx bits descriptions in the TAMP_ATCRx (x = 1, 2) registers.

Every two CK_ATPER cycles (CK_ATPER = 2ATPER × CK_ATPRE), TAMP_OUTy output pin provides a value provided by a pseudo random number generator (PRNG). After outputting this value, the TAMP_OUTy pin outputs its opposite value one CK_ATPER cycle after.
PRNG is consumed by the selected tamper outputs at a different frequency depending on the number of selected tamper outputs. The number of selected outputs depends on TAMPxAM, TAMPxE, ATOSEL and ATOSHARE.

- When only 1 output is selected: PRNG is consumed every 16 CK_ATPER periods.
- When 2 outputs are selected: PRNG is consumed every 8 CK_ATPER periods.
- When 3 or 4 outputs are selected: PRNG is consumed every 4 CK_ATPER periods.

### Table 491. Active tamper output change period

<table>
<thead>
<tr>
<th>ATCKSEL[3:0]</th>
<th>CK_ATPRE frequency</th>
<th>ATPER[2:0]</th>
<th>Tamper output change (CK_ATPER) frequency</th>
<th>Tamper output change period (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>f_RTCCLK</td>
<td>0x0</td>
<td>f_RTCCLK</td>
<td>0.030</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x1</td>
<td>f_RTCCLK/2</td>
<td>0.061</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x2</td>
<td>f_RTCCLK/4</td>
<td>0.122</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x3</td>
<td>f_RTCCLK/8</td>
<td>0.244</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x4</td>
<td>f_RTCCLK/16</td>
<td>0.488</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x5</td>
<td>f_RTCCLK/32</td>
<td>0.977</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x6</td>
<td>f_RTCCLK/64</td>
<td>1.953</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x7</td>
<td>f_RTCCLK/128</td>
<td>3.906</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x7</td>
<td>f_RTCCLK/128</td>
<td>0x0</td>
<td>f_RTCCLK/128</td>
<td>3.906</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x1</td>
<td>f_RTCCLK/256</td>
<td>7.8125</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x2</td>
<td>f_RTCCLK/512</td>
<td>15.625</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x3</td>
<td>f_RTCCLK/1024</td>
<td>31.250</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x4</td>
<td>f_RTCCLK/2048</td>
<td>62.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x5</td>
<td>f_RTCCLK/4096</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x6</td>
<td>f_RTCCLK/8192</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x7</td>
<td>f_RTCCLK/16384</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xB</td>
<td>f_RTCCLK/2048(2)</td>
<td>0x0</td>
<td>f_RTCCLK/2048</td>
<td>62.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x1</td>
<td>f_RTCCLK/4096</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x2</td>
<td>f_RTCCLK/8192</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x3</td>
<td>f_RTCCLK/16384</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x4</td>
<td>f_RTCCLK/32768</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x5</td>
<td>f_RTCCLK/65536</td>
<td>2000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x6</td>
<td>f_RTCCLK/131072</td>
<td>4000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x7</td>
<td>f_RTCCLK/262144</td>
<td>8000</td>
</tr>
</tbody>
</table>

1. Assuming f_RTCCLK = 32768 Hz.
2. This setting requires that (PREDIV_A+1) = 128 and (PREDIV_S+1) is a multiple of 16.
When 5 or more outputs are selected: PRNG is consumed every 2 CK_ATPER periods.
The PRNG needs minimum 9 CK_ATPRE cycles to output a new value. Consequently the minimum ATPER values for correct functionality are provided in the table below:

Table 492. Minimum ATPER value

<table>
<thead>
<tr>
<th>Number of selected outputs</th>
<th>Minimum ATPER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3 or 4</td>
<td>2</td>
</tr>
<tr>
<td>5 or more</td>
<td>3</td>
</tr>
</tbody>
</table>

The TAMP_INx pin is externally connected to TAMP_OUTy pin. The comparison is made between TAMP_OUTy output value and TAMP_INx received value, every CK_ATPRE cycle. In case a comparison mismatch occurs, the TAMPxF bit is set in the TAMP_SR register.

As an example, TAMP_OUT1 can be used for comparison with TAMP_IN1 and TAMP_IN2 by configuring and enabling both TAMP1 and TAMP2 in active mode, with ATOSHARE = 1, ATOSEL1 = 000 and ATOSEL2 = 000.

The active tamper can be combined with input filtering when FLTEN = 1. In this case, the tamper is detected only when 2 comparisons are false, in 4 consecutive comparison samples.

As illustrated in Figure 620, if FLTEN = 0, any mismatch between the TAMP_OUTy output and the associated TAMP_INx input when the latter is sampled generates a tamper. This is the case in all three examples (a), (b) and (c).

If FLTEN = 1, example (a) does not generate a tamper, since only one mismatch is detected in four consecutive comparisons. In example (b), a tamper is generated since two successive mismatches are detected. Example (c) also generates a tamper, since two mismatches occur in four consecutive comparisons, even though the mismatches do not occur on successive samples.
Setting FLTEN = 1 avoids unwanted detection of tampers due to glitches, bounce or transitory states on the TAMP_INx inputs, by ignoring single pulses which are shorter than one period of CK_ATPRE, programmed in the ATCKSEL field of the TAMP_ATCR1 register. The minimum filtered pulse width is listed in Table 493 for each possible setting of ATCKSEL, assuming f_RTCCLK = 32.768 kHz.

### Table 493. Active tamper filtered pulse duration

<table>
<thead>
<tr>
<th>ATCKSEL[3:0]</th>
<th>CK_ATPRE frequency</th>
<th>Minimum filtered pulse width (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>f_RTCCLK</td>
<td>0.030</td>
</tr>
<tr>
<td>0x1</td>
<td>f_RTCCLK/2</td>
<td>0.061</td>
</tr>
<tr>
<td>0x2</td>
<td>f_RTCCLK/4</td>
<td>0.122</td>
</tr>
<tr>
<td>0x3</td>
<td>f_RTCCLK/8</td>
<td>0.244</td>
</tr>
<tr>
<td>0x4</td>
<td>f_RTCCLK/16</td>
<td>0.488</td>
</tr>
<tr>
<td>0x5</td>
<td>f_RTCCLK/32</td>
<td>0.977</td>
</tr>
<tr>
<td>0x6</td>
<td>f_RTCCLK/64</td>
<td>1.953</td>
</tr>
<tr>
<td>0x7</td>
<td>f_RTCCLK/128</td>
<td>3.906</td>
</tr>
<tr>
<td>0x8</td>
<td>f_RTCCLK/2048</td>
<td>62.500(1)</td>
</tr>
</tbody>
</table>

1. This setting requires that \((\text{PREDIV}_A+1) = 128\) and \((\text{PREDIV}_S+1)\) is a multiple of 16.

**Note:** Multiple pulses which are shorter than one CK_ATPRE period may nevertheless cause a tamper if they result in two mismatches in four consecutive comparisons.

**Caution:**

Refer to section **Section : Calendar initialization and configuration**.

Refer also to **RTC alarm A subsecond register (RTC_ALRMASSR)**, **RTC alarm B subsecond register (RTC_ALRMBSSR)**, **RTC alarm A binary mode register (RTC_ALRABINR)** and **RTC alarm B binary mode register (RTC_ALRBBINR)** in case RTC binary mode is used in conjunction with ATCKSEL[3] = 1.

**Caution:**
Caution: The active tamper detection is no more functional in case of calendar overflow when ATCKSEL[3] = 1. It is mandatory to enable the internal tamper 5 on calendar overflow to ensure tamper protection.

The pseudo-random generator must be initialized with a seed. This is done by writing consecutively four 32-bit random values in the TAMP_ATSEEDR register. Programming the seed automatically sends it to the PRNG. As long as the new seed is transferred and elaborated by the PRNG, the SEEDF bit is set in the TAMP_ATOR and it is not allowed to switch off the TAMP APB clock. The duration of the elaboration is up to 184 APB clock cycles after the forth seed is written. Consequently, after writing a new seed, the user must wait until SEEDF is cleared before entering low-power modes.

The active tamper outputs are activated only after the first seed is written and the elaboration is completed. Then new seeds can be written and elaborated during active tamper activity.

**Active tamper initialization**

Here is the software procedure to initialize the active tampers after system reset:
Read INITS in TAMP_ATOR register.

- If INITS = 0x0 (initialization was not done):
  a) Write TAMP_ATCR to configure Active tamper clock, filter and output sharing if any, and active mode.
  b) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  c) Write SEED by writing four times in the TAMP_ATSEEDR.
  d) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.

- If INITS = 0x1 (initialization already done):
  No initialization. To increase randomness a new SEED should be provided regularly. When a new SEED is provided, wait until SEEDF = 0 before entering a low-power mode which switches off the TAMP APB clock.

- In case the tampers are disabled by software, and re-enabled afterwards, the SEED must be written after enabling tampers:
  a) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  b) Write SEED by writing four times in the TAMP_ATSEEDR.
  c) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.

### 47.4 TAMP low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. TAMP interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>No effect on all features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. Tamper events cause the device to exit the Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>No effect on all features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. Tamper events cause the device to exit the Standby mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Functional in all low-power modes</th>
<th>Functional in VBAT mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMP_IN[8:1]</td>
<td>Yes(^{(1)})</td>
<td>Yes(^{(1)})</td>
</tr>
<tr>
<td>TAMP_OUT[8:1]</td>
<td>Yes(^{(2)})</td>
<td>Yes(^{(3)})</td>
</tr>
</tbody>
</table>

1. Only PC13, PI8, PA0, PA1 and PA2 are functional in Standby and VBAT modes.
2. Only PC13, PA1 and PI8 are functional in Standby mode.
3. Only PC13, PA1 and PI8 are functional in VBAT mode.
47.5 TAMP interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration (TAMPSEC).

Table 496. Interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag(1)</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from low-power modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMP</td>
<td>TAMPx</td>
<td>TAMPxF</td>
<td></td>
<td>Write 1 in CTAMPxF</td>
<td>Yes(3)</td>
</tr>
<tr>
<td></td>
<td>Internal tamper</td>
<td>ITAMPy</td>
<td>ITAMPyl</td>
<td>Write 1 in CITAAPPy</td>
<td>Yes(3)</td>
</tr>
</tbody>
</table>

1. The event flags are in the TAMP_SR register.
2. The number of tampers and internal tampers events depend on products.
3. Refer to Table 494: Effect of low-power modes on TAMP for more details about available features in the low-power modes.

47.6 TAMP registers

Refer to Section 1.2 of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

47.6.1 TAMP control register 1 (TAMP_CR1)

This register can be protected against nonsecure access. Refer to Section 47.3.5: TAMP secure protection modes.

This register can be protected against non-privileged access. Refer to Section 47.3.7: TAMP privilege protection modes.

Address offset: 0x00
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td></td>
<td>rw</td>
<td></td>
<td>rw</td>
<td></td>
<td>rw</td>
<td></td>
<td>rw</td>
<td></td>
<td>rw</td>
<td></td>
<td>rw</td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bit 30 ITAMP15E: Internal tamper 15 enable
0: Internal tamper 15 disabled.
1: Internal tamper 15 enabled.

Bit 29 Reserved, must be kept at reset value.
Bit 28  **ITAMP13E**: Internal tamper 13 enable
   0: Internal tamper 13 disabled.
   1: Internal tamper 13 enabled.

Bit 27  **ITAMP12E**: Internal tamper 12 enable
   0: Internal tamper 12 disabled.
   1: Internal tamper 12 enabled.

Bit 26  **ITAMP11E**: Internal tamper 11 enable
   0: Internal tamper 11 disabled.
   1: Internal tamper 11 enabled.

Bit 25  Reserved, must be kept at reset value.

Bit 24  **ITAMP9E**: Internal tamper 9 enable
   0: Internal tamper 9 disabled.
   1: Internal tamper 9 enabled.

Bit 23  **ITAMP8E**: Internal tamper 8 enable
   0: Internal tamper 8 disabled.
   1: Internal tamper 8 enabled.

Bit 22  **ITAMP7E**: Internal tamper 7 enable
   0: Internal tamper 7 disabled.
   1: Internal tamper 7 enabled

Bit 21  **ITAMP6E**: Internal tamper 6 enable
   0: Internal tamper 6 disabled.
   1: Internal tamper 6 enabled.

Bit 20  **ITAMP5E**: Internal tamper 5 enable
   0: Internal tamper 5 disabled.
   1: Internal tamper 5 enabled.

Bit 19  **ITAMP4E**: Internal tamper 4 enable
   0: Internal tamper 4 disabled.
   1: Internal tamper 4 enabled.

Bit 18  **ITAMP3E**: Internal tamper 3 enable
   0: Internal tamper 3 disabled.
   1: Internal tamper 3 enabled.

Bit 17  **ITAMP2E**: Internal tamper 2 enable
   0: Internal tamper 2 disabled.
   1: Internal tamper 2 enabled.

Bit 16  **ITAMP1E**: Internal tamper 1 enable
   0: Internal tamper 1 disabled.
   1: Internal tamper 1 enabled.

Bits 15:8  Reserved, must be kept at reset value.

Bit 7  **TAMP8E**: Tamper detection on TAMP_IN8 enable
   0: Tamper detection on TAMP_IN8 is disabled.
   1: Tamper detection on TAMP_IN8 is enabled.

Bit 6  **TAMP7E**: Tamper detection on TAMP_IN7 enable
   0: Tamper detection on TAMP_IN7 is disabled.
   1: Tamper detection on TAMP_IN7 is enabled.
Bit 5 **TAMP6E**: Tamper detection on TAMP_IN6 enable\(^{(1)}\)

- 0: Tamper detection on TAMP_IN6 is disabled.
- 1: Tamper detection on TAMP_IN6 is enabled.

Bit 4 **TAMP5E**: Tamper detection on TAMP_IN5 enable\(^{(1)}\)

- 0: Tamper detection on TAMP_IN5 is disabled.
- 1: Tamper detection on TAMP_IN5 is enabled.

Bit 3 **TAMP4E**: Tamper detection on TAMP_IN4 enable\(^{(1)}\)

- 0: Tamper detection on TAMP_IN4 is disabled.
- 1: Tamper detection on TAMP_IN4 is enabled.

Bit 2 **TAMP3E**: Tamper detection on TAMP_IN3 enable\(^{(1)}\)

- 0: Tamper detection on TAMP_IN3 is disabled.
- 1: Tamper detection on TAMP_IN3 is enabled.

Bit 1 **TAMP2E**: Tamper detection on TAMP_IN2 enable\(^{(1)}\)

- 0: Tamper detection on TAMP_IN2 is disabled.
- 1: Tamper detection on TAMP_IN2 is enabled.

Bit 0 **TAMP1E**: Tamper detection on TAMP_IN1 enable\(^{(1)}\)

- 0: Tamper detection on TAMP_IN1 is disabled.
- 1: Tamper detection on TAMP_IN1 is enabled.

1. Tamper detection mode (selected with TAMP_FLTCR, TAMP_ATCR1, TAMP_ATCR2 registers and TAMPxTRG bits in TAMP_CR2), must be configured before enabling the tamper detection.

### 47.6.2 TAMP control register 2 (TAMP_CR2)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x04  
Backup domain reset value: 0x0000 0000  
System reset: not affected

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>30</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>29</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>28</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>27</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>26</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>25</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>24</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>23</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>22</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>21</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>20</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>19</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>18</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>17</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
<tr>
<td>16</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
</tbody>
</table>

---

RM0481 Rev 1 2023/3119
Bit 31  **TAMP8TRG**: Active level for tamper 8 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 tamper 8 input staying low triggers a tamper detection event.
- 1: If TAMPFLT = 00 tamper 8 input rising edge triggers a tamper detection event.

Bit 30  **TAMP7TRG**: Active level for tamper 7 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 tamper 7 input staying low triggers a tamper detection event.
- 1: If TAMPFLT = 00 tamper 7 input rising edge triggers a tamper detection event.

Bit 29  **TAMP6TRG**: Active level for tamper 6 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 tamper 6 input staying low triggers a tamper detection event.
- 1: If TAMPFLT = 00 tamper 6 input rising edge triggers a tamper detection event.

Bit 28  **TAMP5TRG**: Active level for tamper 5 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 tamper 5 input staying low triggers a tamper detection event.
- 1: If TAMPFLT = 00 tamper 5 input rising edge triggers a tamper detection event.

Bit 27  **TAMP4TRG**: Active level for tamper 4 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 tamper 4 input staying low triggers a tamper detection event.
- 1: If TAMPFLT = 00 tamper 4 input rising edge triggers a tamper detection event.

Bit 26  **TAMP3TRG**: Active level for tamper 3 input
- 0: If TAMPFLT ≠ 00 tamper 3 input staying low triggers a tamper detection event.
- 1: If TAMPFLT = 00 tamper 3 input rising edge triggers a tamper detection event.

Bit 25  **TAMP2TRG**: Active level for tamper 2 input
- 0: If TAMPFLT ≠ 00 tamper 2 input staying low triggers a tamper detection event.
- 1: If TAMPFLT = 00 tamper 2 input rising edge triggers a tamper detection event.

Bit 24  **TAMP1TRG**: Active level for tamper 1 input
- 0: If TAMPFLT ≠ 00 tamper 1 input staying low triggers a tamper detection event.
- 1: If TAMPFLT = 00 tamper 1 input rising edge triggers a tamper detection event.

Bit 23  **BKERASE**: Backup registers and device secrets\(^{(1)}\) erase
- Writing “1” to this bit reset the backup registers and device secrets\(^{(1)}\). Writing 0 has no effect.
- This bit is always read as 0.

Bit 22  **BKBLOCK**: Backup registers and device secrets\(^{(1)}\) access blocked
- 0: backup registers and device secrets\(^{(1)}\) can be accessed if no tamper flag is set
- 1: backup registers and device secrets\(^{(1)}\) cannot be accessed

Bits 21:19  Reserved, must be kept at reset value.
Bit 18 **TAMP3MSK**: Tamper 3 mask
0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.
1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers and device secrets\(^1\) are not erased.
*The tamper 3 interrupt must not be enabled when TAMP3MSK is set.*

Bit 17 **TAMP2MSK**: Tamper 2 mask
0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.
1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers and device secrets\(^1\) are not erased.
*The tamper 2 interrupt must not be enabled when TAMP2MSK is set.*

Bit 16 **TAMP1MSK**: Tamper 1 mask
0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.
1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers and device secrets\(^1\) are not erased.
*The tamper 1 interrupt must not be enabled when TAMP1MSK is set.*

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TAMP8POM**: Tamper 8 potential mode
0: Tamper 8 event detection is in confirmed mode\(^1\).
1: Tamper 8 event detection is in potential mode\(^2\).

Bit 6 **TAMP7POM**: Tamper 7 potential mode
0: Tamper 7 event detection is in confirmed mode\(^1\).
1: Tamper 7 event detection is in potential mode\(^2\).

Bit 5 **TAMP6POM**: Tamper 6 potential mode
0: Tamper 6 event detection is in confirmed mode\(^1\).
1: Tamper 6 event detection is in potential mode\(^2\).

Bit 4 **TAMP5POM**: Tamper 5 potential mode
0: Tamper 5 event detection is in confirmed mode\(^1\).
1: Tamper 5 event detection is in potential mode\(^2\).

Bit 3 **TAMP4POM**: Tamper 4 potential mode
0: Tamper 4 event detection is in confirmed mode\(^1\).
1: Tamper 4 event detection is in potential mode\(^2\).

Bit 2 **TAMP3POM**: Tamper 3 potential mode
0: Tamper 3 event detection is in confirmed mode\(^1\).
1: Tamper 3 event detection is in potential mode\(^2\).

Bit 1 **TAMP2POM**: Tamper 2 potential mode
0: Tamper 2 event detection is in confirmed mode\(^1\).
1: Tamper 2 event detection is in potential mode\(^2\).

Bit 0 **TAMP1POM**: Tamper 1 potential mode
0: Tamper 1 event detection is in confirmed mode\(^1\).
1: Tamper 1 event detection is in potential mode\(^2\).

1. The effects of tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in Table 489: TAMP interconnection.
2. The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in Table 489: TAMP interconnection.
### 47.6.3 TAMP control register 3 (TAMP_CR3)

This register can be protected against nonsecure access. Refer to Section 47.3.5: TAMP secure protection modes.

This register can be protected against non-privileged access. Refer to Section 47.3.7: TAMP privilege protection modes.

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| 15POM | ITAMP15POM | Res | 13POM | ITAMP13POM | 12POM | ITAMP12POM | Res |  9POM | ITAMP9POM |  8POM | ITAMP8POM |  7POM | ITAMP7POM | Res |  5POM | ITAMP5POM |  4POM | ITAMP4POM |  3POM | ITAMP3POM | Res |  1POM | ITAMP1POM |
|-------|------------|-----|-------|------------|-------|------------|-----|-------|------------|-------|------------|-------|------------|-----|-------|------------|-------|------------|-----|-------|------------|
| rw    | rw         | rw  | rw    | rw         | rw    | rw         | rw  | rw    | rw         | rw    | rw         | rw    | rw         | rw  | rw    | rw         | rw    | rw         | rw  | rw    | rw         |

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ITAMP15POM**: Internal tamper 15 potential mode
- 0: Internal tamper 15 event detection is in confirmed mode\(^{(1)}\).
- 1: Internal tamper 15 event detection is in potential mode\(^{(2)}\).

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ITAMP13POM**: Internal tamper 13 potential mode
- 0: Internal tamper 13 event detection is in confirmed mode\(^{(1)}\).
- 1: Internal tamper 13 event detection is in potential mode\(^{(2)}\).

Bit 11 **ITAMP12POM**: Internal tamper 12 potential mode
- 0: Internal tamper 12 event detection is in confirmed mode\(^{(1)}\).
- 1: Internal tamper 12 event detection is in potential mode\(^{(2)}\).

Bit 10 **ITAMP11POM**: Internal tamper 11 potential mode
- 0: Internal tamper 11 event detection is in confirmed mode\(^{(1)}\).
- 1: Internal tamper 11 event detection is in potential mode\(^{(2)}\).

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ITAMP9POM**: Internal tamper 9 potential mode
- 0: Internal tamper 9 event detection is in confirmed mode\(^{(1)}\).
- 1: Internal tamper 9 event detection is in potential mode\(^{(2)}\).

Bit 7 **ITAMP8POM**: Internal tamper 8 potential mode
- 0: Internal tamper 8 event detection is in confirmed mode\(^{(1)}\).
- 1: Internal tamper 8 event detection is in potential mode\(^{(2)}\).

Bit 6 **ITAMP7POM**: Internal tamper 7 potential mode
- 0: Internal tamper 7 event detection is in confirmed mode\(^{(1)}\).
- 1: Internal tamper 7 event detection is in potential mode\(^{(2)}\).
Bit 5 **ITAMP6POM**: Internal tamper 6 potential mode
0: Internal tamper 6 event detection is in confirmed mode(1).
1: Internal tamper 6 event detection is in potential mode(2).

Bit 4 **ITAMP5POM**: Internal tamper 5 potential mode
0: Internal tamper 5 event detection is in confirmed mode(1).
1: Internal tamper 5 event detection is in potential mode(2).

Bit 3 **ITAMP4POM**: Internal tamper 4 potential mode
0: Internal tamper 4 event detection is in confirmed mode(1).
1: Internal tamper 4 event detection is in potential mode(2).

Bit 2 **ITAMP3POM**: Internal tamper 3 potential mode
0: Internal tamper 3 event detection is in confirmed mode(1).
1: Internal tamper 3 event detection is in potential mode(2).

Bit 1 **ITAMP2POM**: Internal tamper 2 potential mode
0: Internal tamper 2 event detection is in confirmed mode(1).
1: Internal tamper 2 event detection is in potential mode(2).

Bit 0 **ITAMP1POM**: Internal tamper 1 potential mode
0: Internal tamper 1 event detection is in confirmed mode(1).
1: Internal tamper 1 event detection is in potential mode(2).

1. The effects of internal tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in Table 489: TAMP interconnection.
2. The effects of internal tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in Table 489: TAMP interconnection.

### 47.6.4 TAMP filter control register (TAMP_FLTCR)

This register can be protected against nonsecure access. Refer to Section 47.3.5: TAMP secure protection modes.

This register can be protected against non-privileged access. Refer to Section 47.3.7: TAMP privilege protection modes.

Address offset: 0x0C

Backup domain reset value: 0x0000 0000

System reset: not affected
Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **TAMPUDIS**: TAMP\_INx pull-up disable
This bit determines if each of the TAMPx pins are precharged before each sample.
0: Precharge TAMP\_INx pins before sampling (enable internal pull-up)
1: Disable precharge of TAMP\_INx pins.

Bits 6:5 **TAMPPRCH[1:0]**: TAMP\_INx precharge duration
These bits determine the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the TAMP\_INx inputs.
0x0: 1 RTCCLK cycle
0x1: 2 RTCCLK cycles
0x2: 4 RTCCLK cycles
0x3: 8 RTCCLK cycles

Bits 4:3 **TAMPFLT[1:0]**: TAMP\_INx filter count
These bits determine the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a tamper event. TAMPFLT is valid for each of the TAMP\_INx inputs.
0x0: Tamper event is activated on edge of TAMP\_INx input transitions to the active level (no internal pull-up on TAMP\_INx input).
0x1: Tamper event is activated after 2 consecutive samples at the active level.
0x2: Tamper event is activated after 4 consecutive samples at the active level.
0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 2:0 **TAMPFREQ[2:0]**: Tamper sampling frequency
Determines the frequency at which each of the TAMP\_INx inputs are sampled.
0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)
0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)
0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)
0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)
0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)
0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)
0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)
0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

**Note:** This register concerns only the tamper inputs in passive mode.

47.6.5 **TAMP active tamper control register 1 (TAMP\_ATCR1)**

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes.*

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes.*
Address offset: 0x10
Backup domain reset value: 0x0007 0000
System reset: not affected

Bit 31 **FLTEN**: Active tamper filter enable
0: Active tamper filtering disable
1: Active tamper filtering enable: a tamper event is detected when 2 comparison mismatches occur out of 4 consecutive samples.

Bit 30 **ATOSHARE**: Active tamper output sharing
0: Each active tamper input TAMP_INi is compared with its dedicated output TAMP_OUTi
1: Each active tamper input TAMP_INi is compared with TAMPOUTSELi defined by ATOSELi bits.

Bits 29:27 Reserved, must be kept at reset value.

Bits 26:24 **ATPER[2:0]**: Active tamper output change period
The tamper output is changed every CK_ATPER = (2^ATPER x CK_ATPRE) cycles. Refer to Table 492: Minimum ATPER value.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ATCKSEL[3:0]**: Active tamper RTC asynchronous prescaler clock selection
These bits selects the RTC asynchronous prescaler stage output. The selected clock is CK_ATPRE.
0000: RTCCLK is selected
0001: RTCCLK/2 is selected
0010: RTCCLK/4 is selected
0011: RTCCLK/8 is selected
0100: RTCCLK/16 is selected
0101: RTCCLK/32 is selected
0110: RTCCLK/64 is selected
0111: RTCCLK/128 is selected
1011: RTCCLK/2048 is selected when (PREDIV_A+1) = 128 and (PREDIV_S+1) is a multiple of 16.
Others: Reserved

Note: These bits can be written only when all active tampers are disabled. The write protection remains for up to 1.5 CK_ATPRE cycles after all the active tampers are disable.
Bits 15:14 **ATOSEL4[1:0]**: Active tamper shared output 4 selection
- 00: TAMPOUTSEL4 = TAMP_OUT1
- 01: TAMPOUTSEL4 = TAMP_OUT2
- 10: TAMPOUTSEL4 = TAMP_OUT3
- 11: TAMPOUTSEL4 = TAMP_OUT4

If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bits 13:12 **ATOSEL3[1:0]**: Active tamper shared output 3 selection
- 00: TAMPOUTSEL3 = TAMP_OUT1
- 01: TAMPOUTSEL3 = TAMP_OUT2
- 10: TAMPOUTSEL3 = TAMP_OUT3
- 11: TAMPOUTSEL3 = TAMP_OUT4

If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bits 11:10 **ATOSEL2[1:0]**: Active tamper shared output 2 selection
- 00: TAMPOUTSEL2 = TAMP_OUT1
- 01: TAMPOUTSEL2 = TAMP_OUT2
- 10: TAMPOUTSEL2 = TAMP_OUT3
- 11: TAMPOUTSEL2 = TAMP_OUT4

If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bits 9:8 **ATOSEL1[1:0]**: Active tamper shared output 1 selection
- 00: TAMPOUTSEL1 = TAMP_OUT1
- 01: TAMPOUTSEL1 = TAMP_OUT2
- 10: TAMPOUTSEL1 = TAMP_OUT3
- 11: TAMPOUTSEL1 = TAMP_OUT4

If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bit 7 **TAMP8AM**: Tamper 8 active mode
- 0: Tamper 8 detection mode is passive.
- 1: Tamper 8 detection mode is active.

Bit 6 **TAMP7AM**: Tamper 7 active mode
- 0: Tamper 7 detection mode is passive.
- 1: Tamper 7 detection mode is active.

Bit 5 **TAMP6AM**: Tamper 6 active mode
- 0: Tamper 6 detection mode is passive.
- 1: Tamper 6 detection mode is active.

Bit 4 **TAMP5AM**: Tamper 5 active mode
- 0: Tamper 5 detection mode is passive.
- 1: Tamper 5 detection mode is active.

Bit 3 **TAMP4AM**: Tamper 4 active mode
- 0: Tamper 4 detection mode is passive.
- 1: Tamper 4 detection mode is active.
Bit 2 **TAMP3AM**: Tamper 3 active mode
0: Tamper 3 detection mode is passive.
1: Tamper 3 detection mode is active.

Bit 1 **TAMP2AM**: Tamper 2 active mode
0: Tamper 2 detection mode is passive.
1: Tamper 2 detection mode is active.

Bit 0 **TAMP1AM**: Tamper 1 active mode
0: Tamper 1 detection mode is passive.
1: Tamper 1 detection mode is active.

**Note:** Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

### 47.6.6 **TAMP active tamper seed register (TAMP_ATSEEDR)**

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x14

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:0 **SEED[31:0]**: Pseudo-random generator seed value
This register must be written four times with 32-bit values to provide the 128-bit seed to the PRNG. Writing to this register automatically sends the seed value to the PRNG.
47.6.7  TAMP active tamper output register (TAMP_ATOR)

This register can be protected against nonsecure access. Refer to Section 47.3.5: TAMP secure protection modes.

This register can be protected against non-privileged access. Refer to Section 47.3.7: TAMP privilege protection modes.

Address offset: 0x18
Backup domain reset value: 0x0000 0000
System reset: not affected, except for SEEDF which is reset to 0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INITS</td>
<td>SEEDF</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>rés</td>
<td>rés</td>
<td>rés</td>
<td>rés</td>
<td>rés</td>
<td>rés</td>
<td>rés</td>
<td>rés</td>
<td>rés</td>
<td>rés</td>
<td>rés</td>
<td>rés</td>
<td>rés</td>
<td>rés</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bit 15  INITS. Active tamper initialization status
This flag is set by hardware when the PRNG has absorbed the first 128-bit seed, meaning that the enabled active tampers are functional. This flag is cleared when the active tampers are disabled.

Bit 14  SEEDF: Seed running flag
This flag is set by hardware when a new seed is written in the TAMP_ATSEEDR. It is cleared by hardware when the PRNG has absorbed this new seed, and by system reset. The TAMP APB clock must not be switched off as long as SEEDF is set.

Bits 13:8  Reserved, must be kept at reset value.

Bits 7:0  PRNG[7:0]: Pseudo-random generator value
This field provides the values of the PRNG output. Because of potential inconsistencies due to synchronization delays, PRNG must be read at least twice. The read value is correct if it is equal to previous read value.
This field can only be read when the APB is in secure mode.

47.6.8  TAMP active tamper control register 2 (TAMP_ATCR2)

This register can be protected against nonsecure access. Refer to Section 47.3.5: TAMP secure protection modes.

This register can be protected against non-privileged access. Refer to Section 47.3.7: TAMP privilege protection modes.
Address offset: 0x1C
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:29 **ATOSEL8[2:0]**: Active tamper shared output 8 selection
- 000: TAMPOUTSEL8 = TAMP_OUT1
- 001: TAMPOUTSEL8 = TAMP_OUT2
- 010: TAMPOUTSEL8 = TAMP_OUT3
- 011: TAMPOUTSEL8 = TAMP_OUT4
- 100: TAMPOUTSEL8 = TAMP_OUT5
- 101: TAMPOUTSEL8 = TAMP_OUT6
- 110: TAMPOUTSEL8 = TAMP_OUT7
- 111: TAMPOUTSEL8 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bits 28:26 **ATOSEL7[2:0]**: Active tamper shared output 7 selection
- 000: TAMPOUTSEL7 = TAMP_OUT1
- 001: TAMPOUTSEL7 = TAMP_OUT2
- 010: TAMPOUTSEL7 = TAMP_OUT3
- 011: TAMPOUTSEL7 = TAMP_OUT4
- 100: TAMPOUTSEL7 = TAMP_OUT5
- 101: TAMPOUTSEL7 = TAMP_OUT6
- 110: TAMPOUTSEL7 = TAMP_OUT7
- 111: TAMPOUTSEL7 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bits 25:23 **ATOSEL6[2:0]**: Active tamper shared output 6 selection
- 000: TAMPOUTSEL6 = TAMP_OUT1
- 001: TAMPOUTSEL6 = TAMP_OUT2
- 010: TAMPOUTSEL6 = TAMP_OUT3
- 011: TAMPOUTSEL6 = TAMP_OUT4
- 100: TAMPOUTSEL6 = TAMP_OUT5
- 101: TAMPOUTSEL6 = TAMP_OUT6
- 110: TAMPOUTSEL6 = TAMP_OUT7
- 111: TAMPOUTSEL6 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.
Bits 22:20 **ATOSEL5[2:0]**: Active tamper shared output 5 selection
- 000: TAMPOUTSEL5 = TAMP_OUT1
- 001: TAMPOUTSEL5 = TAMP_OUT2
- 010: TAMPOUTSEL5 = TAMP_OUT3
- 011: TAMPOUTSEL5 = TAMP_OUT4
- 100: TAMPOUTSEL5 = TAMP_OUT5
- 101: TAMPOUTSEL5 = TAMP_OUT6
- 110: TAMPOUTSEL5 = TAMP_OUT7
- 111: TAMPOUTSEL5 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 19:17 **ATOSEL4[2:0]**: Active tamper shared output 4 selection
- 000: TAMPOUTSEL4 = TAMP_OUT1
- 001: TAMPOUTSEL4 = TAMP_OUT2
- 010: TAMPOUTSEL4 = TAMP_OUT3
- 011: TAMPOUTSEL4 = TAMP_OUT4
- 100: TAMPOUTSEL4 = TAMP_OUT5
- 101: TAMPOUTSEL4 = TAMP_OUT6
- 110: TAMPOUTSEL4 = TAMP_OUT7
- 111: TAMPOUTSEL4 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 18:17 are the mirror of ATOSEL2[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 16:14 **ATOSEL3[2:0]**: Active tamper shared output 3 selection
- 000: TAMPOUTSEL3 = TAMP_OUT1
- 001: TAMPOUTSEL3 = TAMP_OUT2
- 010: TAMPOUTSEL3 = TAMP_OUT3
- 011: TAMPOUTSEL3 = TAMP_OUT4
- 100: TAMPOUTSEL3 = TAMP_OUT5
- 101: TAMPOUTSEL3 = TAMP_OUT6
- 110: TAMPOUTSEL3 = TAMP_OUT7
- 111: TAMPOUTSEL3 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 15:14 are the mirror of ATOSEL3[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.
Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register. All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write). All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write). A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

### 47.6.9 TAMP secure configuration register (TAMP_SECCFRG)

If TZEN = 1, this register can be written only when the APB access is secure. If TZEN = 0, BKPRWSEC[7:0], BKPWSEC[7:0] and BHKLOCK can be written with nonsecure APB access, and TAMPSEC, CNT1SEC cannot be written.

This register can be globally write-protected, or each bit of this register can be individually write-protected against non-privileged access depending on the TAMP_PRIVCFGR configuration (refer to Section 47.3.7: TAMP privilege protection modes).
Address offset: 0x20
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-20</td>
<td>TAMPSEC: Tamper protection (excluding monotonic counters and backup registers)</td>
</tr>
<tr>
<td>30-29</td>
<td>BHKLOCK: Boot hardware key lock</td>
</tr>
<tr>
<td>28-24</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23-16</td>
<td>BKPWSEC[7:0]: Backup registers write protection offset</td>
</tr>
<tr>
<td>15-8</td>
<td>Protection zone 2 is defined for backup registers from TAMP_BKPyR (y = BKPWSEC - 1, with BKPWSEC &gt; BKPRWSEC):</td>
</tr>
<tr>
<td>7-0</td>
<td>Protection zone 3 is defined for backup registers from TAMP_BKPIR (t = BKPWSEC if BKPWSEC ≥ BKPRWSEC, else t = BKPRWSEC):</td>
</tr>
</tbody>
</table>

Note: If TZEN=0: the protection zone 2 can be read and written with nonsecure access. Note: If BKPWPRIV is set, BKPRWSEC[7:0] can be written only in privileged mode.
Bit 15 **CNT1SEC**: Monotonic counter 1 secure protection
0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is secure or nonsecure.
1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is secure.

Bits 14:8 Reserved, must be kept at reset value.

Bits 7:0 **BKPRWSEC[7:0]**: Backup registers read/write protection offset

**BKPRWSEC** value must be from 0 to 32.

**Protection zone 1** is defined for backup registers from TAMP_BKP0R to TAMP_BKPxR (x = BKPRWSEC-1, with BKPRWSEC ≥ 1).
- if TZEN=1, these backup registers can be read and written only with secure access.
  If BKPRWSEC = 0: there is no protection zone 1.

Refer to Figure 619: Backup registers protection zones.

Note: If TZEN=0: the protection zone 1 can be read and written with nonsecure access.
Note: If BKPRWPRIV is set, BKPRWSEC[7:0] can be written only in privileged mode.

### 47.6.10 TAMP privilege configuration register (TAMP_PRIVCFGFR)

This register can be written only when the APB access is privileged.

When TZEN = 1, this register can be write-protected, or each bit of this register can be individually write-protected against nonsecure access depending on the TAMP_SECCFGFR configuration (refer to Section 47.3.5: TAMP secure protection modes).

Address offset: 0x24

Backup domain reset value: 0x0000 0000

System reset: not affected
### 47.6.11 TAMP interrupt enable register (TAMP_IER)

This register can be protected against nonsecure access. Refer to Section 47.3.5: TAMP secure protection modes.

This register can be protected against non-privileged access. Refer to Section 47.3.7: TAMP privilege protection modes.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>ITAMP15IE</td>
<td>Res.</td>
<td>ITAMP13IE</td>
<td>ITAMP12IE</td>
<td>ITAMP11IE</td>
<td>ITAMP10IE</td>
<td>ITAMP9IE</td>
<td>ITAMP8IE</td>
<td>ITAMP7IE</td>
<td>ITAMP6IE</td>
<td>ITAMP5IE</td>
<td>ITAMP4IE</td>
<td>ITAMP3IE</td>
<td>ITAMP2IE</td>
<td>ITAMP1IE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMP8IE</td>
<td>TAMP7IE</td>
<td>TAMP6IE</td>
<td>TAMP5IE</td>
<td>TAMP4IE</td>
<td>TAMP3IE</td>
<td>TAMP2IE</td>
<td>TAMP1IE</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15IE**: Internal tamper 15 interrupt enable
0: Internal tamper 15 interrupt disabled.
1: Internal tamper 15 interrupt enabled.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13IE**: Internal tamper 13 interrupt enable
0: Internal tamper 13 interrupt disabled.
1: Internal tamper 13 interrupt enabled.
Bit 27  **ITAMP12IE**: Internal tamper 12 interrupt enable
0: Internal tamper 12 interrupt disabled.
1: Internal tamper 12 interrupt enabled.

Bit 26  **ITAMP11IE**: Internal tamper 11 interrupt enable
0: Internal tamper 11 interrupt disabled.
1: Internal tamper 11 interrupt enabled.

Bit 25  Reserved, must be kept at reset value.

Bit 24  **ITAMP9IE**: Internal tamper 9 interrupt enable
0: Internal tamper 9 interrupt disabled.
1: Internal tamper 9 interrupt enabled.

Bit 23  **ITAMP8IE**: Internal tamper 8 interrupt enable
0: Internal tamper 8 interrupt disabled.
1: Internal tamper 8 interrupt enabled.

Bit 22  **ITAMP7IE**: Internal tamper 7 interrupt enable
0: Internal tamper 7 interrupt disabled.
1: Internal tamper 7 interrupt enabled.

Bit 21  **ITAMP6IE**: Internal tamper 6 interrupt enable
0: Internal tamper 6 interrupt disabled.
1: Internal tamper 6 interrupt enabled.

Bit 20  **ITAMP5IE**: Internal tamper 5 interrupt enable
0: Internal tamper 5 interrupt disabled.
1: Internal tamper 5 interrupt enabled.

Bit 19  **ITAMP4IE**: Internal tamper 4 interrupt enable
0: Internal tamper 4 interrupt disabled.
1: Internal tamper 4 interrupt enabled.

Bit 18  **ITAMP3IE**: Internal tamper 3 interrupt enable
0: Internal tamper 3 interrupt disabled.
1: Internal tamper 3 interrupt enabled.

Bit 17  **ITAMP2IE**: Internal tamper 2 interrupt enable
0: Internal tamper 2 interrupt disabled.
1: Internal tamper 2 interrupt enabled.

Bit 16  **ITAMP1IE**: Internal tamper 1 interrupt enable
0: Internal tamper 1 interrupt disabled.
1: Internal tamper 1 interrupt enabled

Bits 15:8  Reserved, must be kept at reset value.

Bit 7  **TAMP8IE**: Tamper 8 interrupt enable
0: Tamper 8 interrupt disabled.
1: Tamper 8 interrupt enabled.

Bit 6  **TAMP7IE**: Tamper 7 interrupt enable
0: Tamper 7 interrupt disabled.
1: Tamper 7 interrupt enabled.

Bit 5  **TAMP6IE**: Tamper 6 interrupt enable
0: Tamper 6 interrupt disabled.
1: Tamper 6 interrupt enabled.
Bit 4 **TAMP5IE**: Tamper 5 interrupt enable
0: Tamper 5 interrupt disabled.
1: Tamper 5 interrupt enabled.

Bit 3 **TAMP4IE**: Tamper 4 interrupt enable
0: Tamper 4 interrupt disabled.
1: Tamper 4 interrupt enabled.

Bit 2 **TAMP3IE**: Tamper 3 interrupt enable
0: Tamper 3 interrupt disabled.
1: Tamper 3 interrupt enabled.

Bit 1 **TAMP2IE**: Tamper 2 interrupt enable
0: Tamper 2 interrupt disabled.
1: Tamper 2 interrupt enabled.

Bit 0 **TAMP1IE**: Tamper 1 interrupt enable
0: Tamper 1 interrupt disabled.
1: Tamper 1 interrupt enabled.

### 47.6.12 TAMP status register (TAMP_SR)

This register can be protected against nonsecure access. Refer to **Section 47.3.5: TAMP secure protection modes**.

This register can be protected against non-privileged access. Refer to **Section 47.3.7: TAMP privilege protection modes**.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>r</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|    |    |

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15F**: Internal tamper 15 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 15.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13F**: Internal tamper 13 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 13.

Bit 27 **ITAMP12F**: Internal tamper 12 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 12.
Bit 26  **ITAMP11F**: Internal tamper 11 flag
       This flag is set by hardware when a tamper detection event is detected on the internal tamper 11.

Bit 25  Reserved, must be kept at reset value.

Bit 24  **ITAMP9F**: Internal tamper 9 flag
       This flag is set by hardware when a tamper detection event is detected on the internal tamper 9.

Bit 23  **ITAMP8F**: Internal tamper 8 flag
       This flag is set by hardware when a tamper detection event is detected on the internal tamper 8.

Bit 22  **ITAMP7F**: Internal tamper 7 flag
       This flag is set by hardware when a tamper detection event is detected on the internal tamper 7.

Bit 21  **ITAMP6F**: Internal tamper 6 flag
       This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.

Bit 20  **ITAMP5F**: Internal tamper 5 flag
       This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19  **ITAMP4F**: Internal tamper 4 flag
       This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.

Bit 18  **ITAMP3F**: Internal tamper 3 flag
       This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17  **ITAMP2F**: Internal tamper 2 flag
       This flag is set by hardware when a tamper detection event is detected on the internal tamper 2.

Bit 16  **ITAMP1F**: Internal tamper 1 flag
       This flag is set by hardware when a tamper detection event is detected on the internal tamper 1.

Bits 15:8  Reserved, must be kept at reset value.

Bit 7  **TAMP8F**: TAMP8 detection flag
       This flag is set by hardware when a tamper detection event is detected on the TAMP8 input.

Bit 6  **TAMP7F**: TAMP7 detection flag
       This flag is set by hardware when a tamper detection event is detected on the TAMP7 input.

Bit 5  **TAMP6F**: TAMP6 detection flag
       This flag is set by hardware when a tamper detection event is detected on the TAMP6 input.

Bit 4  **TAMP5F**: TAMP5 detection flag
       This flag is set by hardware when a tamper detection event is detected on the TAMP5 input.

Bit 3  **TAMP4F**: TAMP4 detection flag
       This flag is set by hardware when a tamper detection event is detected on the TAMP4 input.
47.6.13 TAMP nonsecure masked interrupt status register (TAMP_MISR)

This register can be protected against non-privileged access. Refer to Section 47.3.7: TAMP privilege protection modes.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>ITAMP15MF: internal tamper 15 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware when the internal tamper 15 nonsecure interrupt is raised.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>ITAMP13MF: internal tamper 13 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware when the internal tamper 13 nonsecure interrupt is raised.</td>
</tr>
<tr>
<td>27</td>
<td>ITAMP12MF: internal tamper 12 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware when the internal tamper 12 nonsecure interrupt is raised.</td>
</tr>
<tr>
<td>26</td>
<td>ITAMP11MF: internal tamper 11 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware when the internal tamper 11 nonsecure interrupt is raised.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>ITAMP9MF: internal tamper 9 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware when the internal tamper 9 nonsecure interrupt is raised.</td>
</tr>
<tr>
<td>23</td>
<td>ITAMP8MF: Internal tamper 8 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware when the internal tamper 8 nonsecure interrupt is raised.</td>
</tr>
<tr>
<td>22</td>
<td>ITAMP7MF: Internal tamper 7 tamper nonsecure interrupt masked flag</td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware when the internal tamper 7 nonsecure interrupt is raised.</td>
</tr>
<tr>
<td>21</td>
<td>ITAMP6MF: Internal tamper 6 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td></td>
<td>This flag is set by hardware when the internal tamper 6 nonsecure interrupt is raised.</td>
</tr>
<tr>
<td>Bit</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>20</td>
<td>ITAMP5MF: Internal tamper 5 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td>19</td>
<td>ITAMP4MF: Internal tamper 4 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td>18</td>
<td>ITAMP3MF: Internal tamper 3 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td>17</td>
<td>ITAMP2MF: Internal tamper 2 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td>16</td>
<td>ITAMP1MF: Internal tamper 1 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td>15-8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>7</td>
<td>TAMP8MF: TAMP8 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td>6</td>
<td>TAMP7MF: TAMP7 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td>5</td>
<td>TAMP6MF: TAMP6 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td>4</td>
<td>TAMP5MF: TAMP5 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td>3</td>
<td>TAMP4MF: TAMP4 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td>2</td>
<td>TAMP3MF: TAMP3 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td>1</td>
<td>TAMP2MF: TAMP2 nonsecure interrupt masked flag</td>
</tr>
<tr>
<td>0</td>
<td>TAMP1MF: TAMP1 nonsecure interrupt masked flag</td>
</tr>
</tbody>
</table>

**47.6.14 TAMP secure masked interrupt status register (TAMP_SMISR)**

This register can be protected against nonsecure access. Refer to Section 47.3.5: TAMP secure protection modes.

This register can be protected against non-privileged access. Refer to Section 47.3.7: TAMP privilege protection modes.
Address offset: 0x38
Backup domain reset value: 0x0000 0000
System reset: not affected

Bit 31  Reserved, must be kept at reset value.
Bit 30  ITAMP15MF: internal tamper 15 secure interrupt masked flag
        This flag is set by hardware when the internal tamper 15 secure interrupt is raised.
Bit 29  Reserved, must be kept at reset value.
Bit 28  ITAMP13MF: internal tamper 13 secure interrupt masked flag
        This flag is set by hardware when the internal tamper 13 secure interrupt is raised.
Bit 27  ITAMP12MF: internal tamper 12 secure interrupt masked flag
        This flag is set by hardware when the internal tamper 12 secure interrupt is raised.
Bit 26  ITAMP11MF: internal tamper 11 secure interrupt masked flag
        This flag is set by hardware when the internal tamper 11 secure interrupt is raised.
Bit 25  Reserved, must be kept at reset value.
Bit 24  ITAMP9MF: internal tamper 9 secure interrupt masked flag
        This flag is set by hardware when the internal tamper 9 secure interrupt is raised.
Bit 23  ITAMP8MF: Internal tamper 8 secure interrupt masked flag
        This flag is set by hardware when the internal tamper 8 secure interrupt is raised.
Bit 22  ITAMP7MF: Internal tamper 7 secure interrupt masked flag
        This flag is set by hardware when the internal tamper 7 secure interrupt is raised.
Bit 21  ITAMP6MF: Internal tamper 6 secure interrupt masked flag
        This flag is set by hardware when the internal tamper 6 secure interrupt is raised.
Bit 20  ITAMP5MF: Internal tamper 5 secure interrupt masked flag
        This flag is set by hardware when the internal tamper 5 secure interrupt is raised.
Bit 19  ITAMP4MF: Internal tamper 4 secure interrupt masked flag
        This flag is set by hardware when the internal tamper 4 secure interrupt is raised.
Bit 18  ITAMP3MF: Internal tamper 3 secure interrupt masked flag
        This flag is set by hardware when the internal tamper 3 secure interrupt is raised.
Bit 17  ITAMP2MF: Internal tamper 2 secure interrupt masked flag
        This flag is set by hardware when the internal tamper 2 secure interrupt is raised.
Bit 16  ITAMP1MF: Internal tamper 1 secure interrupt masked flag
        This flag is set by hardware when the internal tamper 1 secure interrupt is raised.

Bits 15:8  Reserved, must be kept at reset value.
Bit 7 **TAMP8MF**: TAMP8 secure interrupt masked flag  
This flag is set by hardware when the tamper 8 secure interrupt is raised.

Bit 6 **TAMP7MF**: TAMP7 secure interrupt masked flag  
This flag is set by hardware when the tamper 7 secure interrupt is raised.

Bit 5 **TAMP6MF**: TAMP6 secure interrupt masked flag  
This flag is set by hardware when the tamper 6 secure interrupt is raised.

Bit 4 **TAMP5MF**: TAMP5 secure interrupt masked flag  
This flag is set by hardware when the tamper 5 secure interrupt is raised.

Bit 3 **TAMP4MF**: TAMP4 secure interrupt masked flag  
This flag is set by hardware when the tamper 4 secure interrupt is raised.

Bit 2 **TAMP3MF**: TAMP3 secure interrupt masked flag  
This flag is set by hardware when the tamper 3 secure interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 secure interrupt masked flag  
This flag is set by hardware when the tamper 2 secure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 secure interrupt masked flag  
This flag is set by hardware when the tamper 1 secure interrupt is raised.

### 47.6.15 TAMP status clear register (TAMP_SCR)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x3C

System reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>CITAMP 15F</td>
<td>Res.</td>
<td>CITAMP 13F</td>
<td>Res.</td>
<td>CITAMP 12F</td>
<td>Res.</td>
<td>CITAMP 11F</td>
<td>Res.</td>
<td>CITAMP 9F</td>
<td>C</td>
<td>ITAMP 8F</td>
<td>C</td>
<td>ITAMP 7F</td>
<td>C</td>
<td>ITAMP 6F</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CITAMP15F**: Clear ITAMP15 detection flag  
Writing 1 in this bit clears the ITAMP15F bit in the TAMP_SR register.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **CITAMP13F**: Clear ITAMP13 detection flag  
Writing 1 in this bit clears the ITAMP13F bit in the TAMP_SR register.

Bit 27 **CITAMP12F**: Clear ITAMP12 detection flag  
Writing 1 in this bit clears the ITAMP12F bit in the TAMP_SR register.
Bit 26  **CITAMP11F**: Clear ITAMP11 detection flag  
Writing 1 in this bit clears the ITAMP11F bit in the TAMP_SR register.

Bit 25  Reserved, must be kept at reset value.

Bit 24  **CITAMP9F**: Clear ITAMP9 detection flag  
Writing 1 in this bit clears the ITAMP9F bit in the TAMP_SR register.

Bit 23  **CITAMP8F**: Clear ITAMP8 detection flag  
Writing 1 in this bit clears the ITAMP8F bit in the TAMP_SR register.

Bit 22  **CITAMP7F**: Clear ITAMP7 detection flag  
Writing 1 in this bit clears the ITAMP7F bit in the TAMP_SR register.

Bit 21  **CITAMP6F**: Clear ITAMP6 detection flag  
Writing 1 in this bit clears the ITAMP6F bit in the TAMP_SR register.

Bit 20  **CITAMP5F**: Clear ITAMP5 detection flag  
Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.

Bit 19  **CITAMP4F**: Clear ITAMP4 detection flag  
Writing 1 in this bit clears the ITAMP4F bit in the TAMP_SR register.

Bit 18  **CITAMP3F**: Clear ITAMP3 detection flag  
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17  **CITAMP2F**: Clear ITAMP2 detection flag  
Writing 1 in this bit clears the ITAMP2F bit in the TAMP_SR register.

Bit 16  **CITAMP1F**: Clear ITAMP1 detection flag  
Writing 1 in this bit clears the ITAMP1F bit in the TAMP_SR register.

Bits 15:8  Reserved, must be kept at reset value.

Bit 7  **CTAMP8F**: Clear TAMP8 detection flag  
Writing 1 in this bit clears the TAMP8F bit in the TAMP_SR register.

Bit 6  **CTAMP7F**: Clear TAMP7 detection flag  
Writing 1 in this bit clears the TAMP7F bit in the TAMP_SR register.

Bit 5  **CTAMP6F**: Clear TAMP6 detection flag  
Writing 1 in this bit clears the TAMP6F bit in the TAMP_SR register.

Bit 4  **CTAMP5F**: Clear TAMP5 detection flag  
Writing 1 in this bit clears the TAMP5F bit in the TAMP_SR register.

Bit 3  **CTAMP4F**: Clear TAMP4 detection flag  
Writing 1 in this bit clears the TAMP4F bit in the TAMP_SR register.

Bit 2  **CTAMP3F**: Clear TAMP3 detection flag  
Writing 1 in this bit clears the TAMP3F bit in the TAMP_SR register.

Bit 1  **CTAMP2F**: Clear TAMP2 detection flag  
Writing 1 in this bit clears the TAMP2F bit in the TAMP_SR register.

Bit 0  **CTAMP1F**: Clear TAMP1 detection flag  
Writing 1 in this bit clears the TAMP1F bit in the TAMP_SR register.
47.6.16 TAMP monotonic counter 1 register (TAMP\_COUNT1R)

This register can be protected against nonsecure access. Refer to Section 47.3.5: TAMP secure protection modes.

This register can be protected against non-privileged access. Refer to Section 47.3.7: TAMP privilege protection modes.

Address offset: 0x040

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUNT[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUNT[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
</tbody>
</table>

Bits 31:0 COUNT[31:0]:

This register is read-only and is incremented by one when a write access is done to this register. This register cannot roll-over and is frozen when reaching the maximum value.

47.6.17 TAMP option register (TAMP\_OR)

This register can be protected against nonsecure access. Refer to Section 47.3.5: TAMP secure protection modes.

This register can be protected against non-privileged access. Refer to Section 47.3.7: TAMP privilege protection modes.

Address offset: 0x50

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
47.6.18  TAMP resources protection configuration register (TAMP_RPCFGR)

This register can be protected against nonsecure access. Refer to Section 47.3.5: TAMP secure protection modes.

This register can be protected against non-privileged access. Refer to Section 47.3.7: TAMP privilege protection modes.

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31  Reserved, must be kept at reset value.

Bits 30:8  Reserved, must be kept at reset value.

Bit 7  Reserved, must be kept at reset value.

Bit 6  Reserved, must be kept at reset value.
Bit 5  Reserved, must be kept at reset value.
Bit 4  Reserved, must be kept at reset value.
Bit 3  Reserved, must be kept at reset value.
Bit 2  Reserved, must be kept at reset value.
Bit 1  Reserved, must be kept at reset value.

Bit 0  **RPCFG0**: Configurable resource 0 protection\(^{(1)}\)

0: Resource 0 is not included in the device secrets protected by TAMP peripheral
1: Resource 0 is included in the device secrets protected by TAMP peripheral

---

**47.6.19  TAMP backup x register (TAMP_BKPxR)**

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x100 + 0x04 * x, (x = 0 to 31)
Backup domain reset value: 0x0000 0000
System reset: not affected

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 **BKP[31:0]**:
The application can write or read data to and from these registers.
In the default (ERASE) configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.
## 47.6.20  TAMP register map

### Table 497. TAMP register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>0x00</th>
<th>0x04</th>
<th>0x08</th>
<th>0x0C</th>
<th>0x10</th>
<th>0x14</th>
<th>0x18</th>
<th>0x1C</th>
<th>0x20</th>
<th>0x24</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>TAMP_CR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x04</td>
<td>TAMP_CR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x08</td>
<td>TAMP_CR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0C</td>
<td>TAMP_FLTCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x10</td>
<td>TAMP_ATCR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x14</td>
<td>TAMP_ATSEEDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x18</td>
<td>TAMP_ATOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x1C</td>
<td>TAMP_ATCR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x20</td>
<td>TAMP_SEC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x24</td>
<td>TAMP_PRIV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 497. TAMP register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>العملية</th>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2C</td>
<td>TAMP_IER</td>
<td>TAMP_IE1 TAMP_IE2 TAMP_IE3 TAMP_IE4 TAMP_IE5 TAMP_IE6 TAMP_IE7 TAMP_IE8 TAMP_IE9 TAMP_IE10 TAMP_IE11 TAMP_IE12 TAMP_IE13 TAMP_IE14 TAMP_IE15</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0x30</td>
<td>TAMP_SR</td>
<td>TAMP_SF1 TAMP_SF2 TAMP_SF3 TAMP_SF4 TAMP_SF5 TAMP_SF6 TAMP_SF7 TAMP_SF8 TAMP_SF9 TAMP_SF10 TAMP_SF11 TAMP_SF12</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0x34</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for the register boundary addresses.
48 Inter-integrated circuit (I2C) interface

48.1 Introduction

The I²C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I²C bus. It provides multimaster capability, and controls all I²C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

The I²C bus interface is also SMBus (system management bus) and PMBus® (power management bus) compatible.

DMA can be used to reduce CPU overload.

48.2 I²C main features

- I²C bus specification rev.03 compatibility:
  - Slave and master modes
  - Multimaster capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask)
  - All 7-bit addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy to use event management
  - Optional clock stretching
  - Software reset

- 1-byte buffer with DMA capability

- Programmable analog and digital noise filters

The following features are also available, depending upon product implementation (see Section 48.3):

- SMBus specification rev.3.0 compatibility:
  - Hardware PEC (packet error checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and device support
  - SMBus alert
  - Timeouts and idle condition detection

- PMBus rev 1.3 standard compatibility

- Independent clock: a choice of independent clock sources allowing the I²C communication speed to be independent from the i2c_pclk reprogramming
• Wake-up from Stop mode on address match

48.3 I2C implementation

Table 498. STM32H563/H573 and STM32H562 I2C implementation

<table>
<thead>
<tr>
<th>I2C features(1)</th>
<th>I2C1</th>
<th>I2C2</th>
<th>I2C3</th>
<th>I2C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-bit addressing mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>10-bit addressing mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Standard-mode (up to 100 kbit/s)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Fast-mode (up to 400 kbit/s)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Independent clock</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Wake-up from stop mode only (no autonomous mode)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SMBus/PMBus</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

1. X = supported.

48.4 I2C functional description

In addition to receiving and transmitting data, this interface converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I^2^C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), Fast-mode (up to 400 kHz) or Fast-mode Plus (up to 1 MHz) I^2^C bus.

This interface can also be connected to an SMBus with data (SDA) and clock (SCL) pins.

If the SMBus feature is supported, the optional SMBus Alert pin (SMBA) is also available.
48.4.1 I2C block diagram

The block diagram of the I2C interface is shown in Figure 621.

**Figure 621. I2C block diagram**

The I2C is clocked by an independent clock source, which allows the I2C to operate independently from the i2c_pclk frequency.
48.4.2 I2C pins and internal signals

Table 499. I2C input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C_SDA</td>
<td>Bidirectional</td>
<td>I2C data</td>
</tr>
<tr>
<td>I2C_SCL</td>
<td>Bidirectional</td>
<td>I2C clock</td>
</tr>
<tr>
<td>I2C_SMBA</td>
<td>Bidirectional</td>
<td>SMBus alert</td>
</tr>
</tbody>
</table>

Table 500. I2C internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i2c_ker_ck</td>
<td>Input</td>
<td>I2C kernel clock, also named I2CCLK in this document</td>
</tr>
<tr>
<td>i2c_pclk</td>
<td>Input</td>
<td>I2C APB clock</td>
</tr>
<tr>
<td>i2c_it</td>
<td>Output</td>
<td>I2C interrupts, refer to Table 513 for the list of interrupt sources</td>
</tr>
<tr>
<td>i2c_rx_dma</td>
<td>Output</td>
<td>I2C receive data DMA request (I2C_RX)</td>
</tr>
<tr>
<td>i2c_tx_dma</td>
<td>Output</td>
<td>I2C transmit data DMA request (I2C_TX)</td>
</tr>
</tbody>
</table>

48.4.3 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck period tI2CCLK must respect the following conditions:
- \( t_{I2CCLK} < \left( t_{LOW} - t_{filters} \right) / 4 \)
- \( t_{I2CCLK} < t_{HIGH} \)

with:
- \( t_{LOW} \): SCL low time and \( t_{HIGH} \): SCL high time
- \( t_{filters} \): when enabled, sum of the delays brought by the analog and by the digital filters.

The digital filter delay is DNF x tI2CCLK.

The i2c_pclk clock period tPCLK must respect the condition:
- \( t_{PCLK} < 4 / 3 t_{SCL} \) (tSCL: SCL period)

Caution: When the I2C kernel is clocked by i2c_pclk, this clock must respect the conditions for tI2CCLK.

48.4.4 Mode selection

The interface can operate in one of the four following modes:
- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver
By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

**Communication flow**

In master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition, and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In slave mode, the interface is capable of recognizing its own addresses (7- or 10-bit), and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contains the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in master mode.

A ninth clock pulse follows the eight clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter (see Figure 622).

![Figure 622. I2C bus protocol](https://example.com/fig622)

Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

### 48.4.5 I2C initialization

**Enabling and disabling the peripheral**

The I2C peripheral clock must be configured and enabled in the clock controller, then the I2C can be enabled by setting the PE bit in the I2C_CR1 register.

When the I2C is disabled (PE = 0), the \( \text{I}^2 \text{C} \) performs a software reset. Refer to Section 48.4.6 for more details.

**Noise filters**

Before enabling the I2C peripheral by setting the PE bit in I2C_CR1 register, the user must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This filter is compliant with the \( \text{I}^2 \text{C} \) specification, which requires the suppression of spikes with pulse width up to 50 ns in Fast-mode and Fast-mode Plus.
user can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than $\text{DNF} \times \text{i2c_ker_ck}$ periods. This allows spikes with a programmable length of 1 to 15 i2c_ker_ck periods to be suppressed.

<table>
<thead>
<tr>
<th></th>
<th>Analog filter</th>
<th>Digital filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse width of</td>
<td>≥ 50 ns</td>
<td>Programmable length, from one to fifteen I2C peripheral clocks</td>
</tr>
<tr>
<td>suppressed spikes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Benefits</td>
<td>Available in Stop mode</td>
<td>Programmable length: extra filtering capability versus standard requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>− Stable length</td>
</tr>
<tr>
<td>Drawbacks</td>
<td>Variation vs. temperature, voltage, process</td>
<td>Wake-up from Stop mode on address match is not available when digital filter is enabled</td>
</tr>
</tbody>
</table>

**Caution:** The filter configuration cannot be changed when the I2C is enabled.

**I2C timings**

The timings must be configured to guarantee correct data hold and setup times, in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C configuration window.
When the SCL falling edge is internally detected, a delay ($t_{SDADEL}$, impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output: $t_{SDADEL} = SDADEL \times t_{PRESC} + t_{I2CCLK}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$.

The total SDA output delay is:

$$t_{SYNCl} + \frac{[SDADEL \times (PRESC + 1) + 1] \times t_{I2CCLK}}{(PRESC + 1) \times t_{I2CCLK}}$$

The delay $t_{SYNCl}$ duration depends upon:

- SCL falling slope
- When enabled, input delay brought by the analog filter: $t_{AF(min)} < t_{AF} < t_{AF(max)}$
- When enabled, input delay brought by the digital filter: $t_{DNF} = DNF \times t_{I2CCLK}$
- Delay due to SCL synchronization to $i2c\_ker\_ck$ clock (two to three $i2c\_ker\_ck$ periods)

To bridge the undefined region of the SCL falling edge, the user must program $SDADEL$ in such a way that:

$$\frac{[t_{(max)} + t_{HD;DAT(min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]]}{(PRESC + 1) \times t_{I2CCLK}} \leq SDADEL$$
SDADEL ≤ \{t_{HD;DAT}(\text{max}) - t_{AF(\text{max})} - [(DNF + 4) \times t_{I2CCLK}] / \{(PRESC + 1) \times t_{I2CCLK}\}

**Note:**
\(t_{AF(\text{min})} / t_{AF(\text{max})}\) are part of the equation only when the analog filter is enabled. Refer to the device datasheet for \(t_{AF}\) values.

The maximum \(t_{HD;DAT}\) can be 3.45 µs for Standard-mode, 0.9 µs for Fast-mode, 0.45 µs for Fast-mode Plus. It must be lower than the maximum of \(t_{VD;DAT}\) by a transition time. This maximum must only be met if the device does not stretch the LOW period (\(t_{LOW}\)) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case. In this case the previous equation becomes:

SDADEL ≤ \{t_{VD;DAT}(\text{max}) - tr(\text{max}) - t_{AF(\text{max})} - [(DNF + 4) \times t_{I2CCLK}] / \{(PRESC + 1) \times t_{I2CCLK}\}.

**Note:** This condition can be violated when NOSTRETCH = 0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL value.

Refer to Table 502 for \(tr\), \(t_{HD;DAT}\), and \(t_{VD;DAT}\) standard values.

- After \(t_{SDADEL}\), or after sending SDA output when the slave had to stretch the clock because the data was not yet written in I2C_TXDR register, SCL line is kept at low level during the setup time. This setup time is \(t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}\), where \(t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}\). \(t_{SCLDEL}\) impacts the setup time \(t_{SU;DAT}\).

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL in such a way that:

\{(tr(\text{max}) + t_{SU;DAT}(\text{min})) / [(PRESC + 1)] \times t_{I2CCLK}] - 1 ≤ SCLDEL

Refer to Table 502 for \(tr\) and \(t_{SU;DAT}\) standard values.

The SDA and SCL transition time values to use are the ones in the application. Using the maximum values from the standard increases the constraints for the SDADEL and SCLDEL calculation, but ensures the feature, whatever the application.

**Note:** At every clock pulse, after SCL falling edge detection, the I2C master or slave stretches SCL low during at least \([(SDADEL + SCLDEL + 1)] \times (PRESC + 1) + 1\) x \(t_{I2CCLK}\), in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDADEL counter is finished, the I2C keeps on stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

If NOSTRETCH = 1 in slave mode, the SCL is not stretched. Consequently, the SDADEL must be programmed so that it guarantees a sufficient setup time.

### Table 502. I²C-SMBus specification data setup and hold times

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
<th>SMBus</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min.</td>
<td>Max</td>
<td>Min.</td>
<td>Max</td>
</tr>
<tr>
<td>(t_{HD;DAT})</td>
<td>Data hold time</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>(t_{VD;DAT})</td>
<td>Data valid time</td>
<td>-</td>
<td>3.45</td>
<td>-</td>
<td>0.9</td>
</tr>
</tbody>
</table>

| Unit |
Additionally, in master mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bit fields in the I2C_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is \( t_{SCLL} = (SCLL + 1) \times t_{PRESC} \) where \( t_{PRESC} = (PRESC + 1) \times t_{I2CCLK} \). \( t_{SCLL} \) impacts the SCL low time \( t_{LOW} \).
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is \( t_{SCLH} = (SCLH + 1) \times t_{PRESC} \), where \( t_{PRESC} = (PRESC + 1) \times t_{I2CCLK} \). \( t_{SCLH} \) impacts the SCL high time \( t_{HIGH} \).

Refer to \textit{I2C master initialization} for more details.

**Caution:** Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOSTRETCH mode must also be configured before enabling the peripheral. Refer to \textit{I2C slave initialization} for more details.

**Caution:** Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.

---

### Table 502. \(^2\text{C}-\text{SMBus} \) specification data setup and hold times (continued)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
<th>SMBus</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{SU,\text{DAT}} )</td>
<td>Data setup time</td>
<td>Min.</td>
<td>Max</td>
<td>Min.</td>
<td>Max</td>
<td>Min.</td>
</tr>
<tr>
<td>( t_r )</td>
<td>Rise time of both SDA and SCL signals</td>
<td>-</td>
<td>1000</td>
<td>-</td>
<td>300</td>
<td>-</td>
</tr>
<tr>
<td>( t_f )</td>
<td>Fall time of both SDA and SCL signals</td>
<td>-</td>
<td>300</td>
<td>-</td>
<td>300</td>
<td>-</td>
</tr>
</tbody>
</table>
48.4.6 Software reset

A software reset can be performed by clearing the PE bit in the I2C_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and communication control bits, as well as status bits, come back to their reset value. The configuration registers are not impacted.

Impacted register bits:
1. I2C_CR2 register: START, STOP, NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

In addition when the SMBus feature is supported:
1. I2C_CR2 register: PECBYTE
2. I2C_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least three APB clock cycles to perform the software reset. This is ensured by the following software sequence:
1. Write PE = 0
2. Check PE = 0
3. Write PE = 1
48.4.7 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch is inserted between the eighth and ninth SCL pulse (before the acknowledge pulse).

Figure 625. Data reception
Transmission

If the I2C_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE = 1, meaning that no data is written yet in I2C_TXDR, SCL line is stretched low until I2C_TXDR is written. The stretch is done after the ninth SCL pulse.

![Figure 626. Data transmission](image)

**Hardware transfer management**

The I2C features an embedded byte counter to manage byte transfer and to close the communication in various modes, such as:

- NACK, STOP and ReSTART generation in master mode
- ACK control in slave receiver mode
- PEC generation/checking when SMBus feature is supported

The byte counter is always used in master mode. By default, it is disabled in slave mode. It can be enabled by software by setting the SBC (slave byte control) bit in the I2C_CR1 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2C_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES is transferred, and an interrupt is generated if TCIE is set. SCL is stretched as long as TCR flag is set. TCR is cleared by software when NBYTES is written to a non-zero value.

When the NBYTES counter is reloaded with the last number of bytes, RELOAD bit must be cleared.
When RELOAD = 0 in master mode, the counter can be used in two modes:

- **Automatic end mode** (AUTOEND = 1 in the I2C_CR2 register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field is transferred.

- **Software end mode** (AUTOEND = 0 in the I2C_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bit field is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit is set in the I2C_CR2 register. This mode must be used when the master wants to send a RESTART condition.

**Caution:** The AUTOEND bit has no effect when the RELOAD bit is set.

<table>
<thead>
<tr>
<th>Function</th>
<th>SBC bit</th>
<th>RELOAD bit</th>
<th>AUTOEND bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Tx/Rx NBYTES + STOP</td>
<td>x</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Master Tx/Rx + NBYTES + RESTART</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Slave Tx/Rx, all received bytes ACKed</td>
<td>0</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Slave Rx with ACK control</td>
<td>1</td>
<td>1</td>
<td>x</td>
</tr>
</tbody>
</table>

### 48.4.8 I2C slave mode

#### I2C slave initialization

To work in slave mode, the user must enable at least one slave address. Registers I2C_OAR1 and I2C_OAR2 are available to program the slave own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default), or in 10-bit addressing mode by setting the OA1MODE bit in the I2C_OAR1 register. OA1 is enabled by setting the OA1EN bit in the I2C_OAR1 register.

- If additional slave addresses are required, the second slave address OA2 can be configured. Up to seven OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address. These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK = 0.

OA2 is enabled by setting the OA2EN bit in the I2C_OAR2 register.

- The general call address is enabled by setting the GCEN bit in the I2C_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, to perform software actions. If the master does not
support clock stretching, the I2C must be configured with NOSTRETCH = 1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled, the user must read the ADDCODE[6:0] bits in the I2C_ISR register to check which address matched. DIR flag must also be checked to know the transfer direction.

**Slave clock stretching (NOSTRETCH = 0)**

In default mode, the I2C slave stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled slave addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCF bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2C_TXDR register, or if the first data byte is not written when the ADDR flag is cleared (TXE = 1). This stretch is released when the data is written to the I2C_TXDR register.
- In reception when the I2C_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2C_RXDR is read.
- When TCR = 1 in Slave Byte Control mode, reload mode (SBC = 1 and RELOAD = 1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] field.
- After SCL falling edge detection, the I2C stretches SCL low during 
  \[ (SDADEL + SCLDEL + 1) \times (PRESC+ 1) + 1 \] x t_{I2CCLK}.

**Slave without clock stretching (NOSTRETCH = 1)**

When NOSTRETCH = 1 in the I2C_CR1 register, the I2C slave does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, it ensures that the OVR status is provided, even for the first data to be transmitted.
- In reception, the data must be read from the I2C_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not, an overrun occurs, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.
Slave byte control mode

To allow byte ACK control in slave reception mode, the Slave byte control mode must be enabled by setting the SBC bit in the I2C_CR1 register. This is required to be compliant with SMBus standards.

The Reload mode must be selected to allow byte ACK control in slave reception mode (RELOAD = 1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and ninth SCL pulses. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent, and the next byte can be received.

NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

Note: The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

Caution: The Slave byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

Figure 627. Slave initialization flow

*SBC must be set to support SMBus features
Slave transmitter

A transmit interrupt status (TXIS) is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C_CR1 register.

The TXIS bit is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C_CR1 register. The slave automatically releases the SCL and SDA lines in order to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C_CR1 register, the STOPF flag is set in the I2C_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to 0. In this case, if TXE = 0 when the slave address is received (ADDR = 1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register by setting the TXE bit in order to program a new data byte.

In Slave byte control mode (SBC = 1), the number of bytes to be transmitted must be programmed in NBYTES in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES.

Caution: When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine, in order to program the first data byte. The first data byte to be sent must be previously programmed in the I2C_TXDR register:

- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2C_TXDR register can be flushed by setting the TXE bit in order to program a new data byte. The STOPF bit must be cleared only after these actions, in order to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

If a TXIS event is needed, (transmit interrupt or transmit DMA request), the user must set the TXIS bit in addition to the TXE bit, in order to generate a TXIS event.
Figure 628. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 0

Slave transmission

Slave initialization

No

I2C_ISR.ADDR = 1?

Yes

Read ADDCODE and DIR in I2C_ISR
Optional: Set I2C_ISR.TXE = 1
Set I2C_ICR.ADDRCF

I2C_ISR.TXIS = 1?

Yes

Write I2C_TXDR.TXDATA

No

SCL stretched
Figure 629. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1

Slave transmission

Slave initialization

I2C_ISR.TXIS = 1?

No

Yes

Write I2C_TXDR.TXDATA

I2C_ISR.STOPF = 1?

No

Yes

Optional: Set I2C_ISR.TXE = 1 and I2C_ISR.TXIS = 1

Set I2C_ICR.STOPCF
Figure 630. Transfer bus diagrams for I2C slave transmitter (mandatory events only)

Example I2C slave transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:

Example I2C slave transmitter 3 bytes without 1st data flush, NOSTRETCH=0:

Example I2C slave transmitter 3 bytes, NOSTRETCH=1:
**Slave receiver**

RXNE is set in I2C_ISR when the I2C_RXDR is full, and generates an interrupt if RXIE is set in I2C_CR1. RXNE is cleared when I2C_RXDR is read.

When a STOP is received and STOPIE is set in I2C_CR1, STOPF is set in I2C_ISR and an interrupt is generated.

---

**Figure 631. Transfer sequence flow for slave receiver with NOSTRETCH = 0**

[Diagram showing the transfer sequence flow for slave receiver with NOSTRETCH = 0]
Figure 632. Transfer sequence flow for slave receiver with NOSTRETCH = 1

Slave reception

Slave initialization

Yes

No

I2C_ISR.RXNE

= 1?

No

Yes

Read I2C_RXDR.RXDATA

I2C_ISR.STOPF

= 1?

No

Yes

Set I2C_ICR.STOPCF

Figure 633. Transfer bus diagrams for I2C slave receiver
(mandatory events only)

Example I2C slave receiver 3 bytes, NOSTRETCH = 0:

Legend

Transmission

Reception

SCL stretch

S Address data1 data2 data3 P

EV1

EV2

EV3

EV4

RXNE

ADDR

RXNE

RXNE

RXNE

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: RXNE ISR: rd data1
EV3: RXNE ISR: rd data2
EV4: RXNE ISR: rd data3

Example I2C slave receiver 3 bytes, NOSTRETCH = 1:

Legend

Transmission

Reception

SCL stretch

S Address data1 data2 data3 A P

EV1

EV2

EV3

RXNE

RXNE

RXNE

RXNE

EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2
EV3: RXNE ISR: rd data3
48.4.9 I2C master mode

I2C master initialization

Before enabling the peripheral, the I2C master clock must be configured by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

A clock synchronization mechanism is implemented in order to support multi-master environment and slave clock stretching.

In order to allow clock synchronization:
- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

The I2C detects its own SCL low level after a tSYNC1 delay depending on the SCL falling edge, SCL input noise filters (analog + digital) and SCL synchronization to the I2CxCLK clock. The I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bits in the I2C_TIMINGR register.

The I2C detects its own SCL high level after a tSYNC2 delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bits in the I2C_TIMINGR register.

Consequently the master clock period is:
\[
\text{t}_{\text{SCL}} = \text{t}_{\text{SYNC1}} + \text{t}_{\text{SYNC2}} + \left\{[(\text{SCLH} + 1) + (\text{SCLL} + 1)] \times (\text{PRESC} + 1) \times \text{t}_{\text{I2CCLK}}\right\}
\]

The duration of tSYNC1 depends upon:
- SCL falling slope
- When enabled, input delay induced by the analog filter
- When enabled, input delay induced by the digital filter: DNF x tI2CCLK
- Delay due to SCL synchronization with i2c_ker_ck clock (two to three i2c_ker_ck periods)

The duration of tSYNC2 depends upon:
- SCL rising slope
- When enabled, input delay induced by the analog filter
- When enabled, input delay induced by the digital filter: DNF x tI2CCLK
- Delay due to SCL synchronization with i2c_ker_ck clock (two to three i2c_ker_ck periods)
Figure 634. Master clock generation

SCL master clock generation

- SCL high level detected
- SCLH counter starts
- SCL released
- SCL driven low
- SCL low level detected
- SCLL counter starts

SCL master clock synchronization

- SCL high level detected
- SCLH counter starts
- SCL low level detected
- SCLL counter starts
- SCL driven low by another device
- SCL driven low by another device
Caution: To be I2C or SMBus compliant, the master clock must respect the timings given in the following table.

Table 504. I2C-SMBus specification clock timings

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
<th>SMBus</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>fSCL</td>
<td>SCL clock frequency</td>
<td>-</td>
<td>100</td>
<td>-</td>
<td>400</td>
<td>-</td>
</tr>
<tr>
<td>tHD:STA</td>
<td>Hold time (repeated) START condition</td>
<td>4.0</td>
<td>0.6</td>
<td>0.26</td>
<td>4.0</td>
<td>-</td>
</tr>
<tr>
<td>tSU:STA</td>
<td>Set-up time for a repeated START condition</td>
<td>4.7</td>
<td>0.6</td>
<td>0.26</td>
<td>4.7</td>
<td>-</td>
</tr>
<tr>
<td>tSU:STO</td>
<td>Set-up time for STOP condition</td>
<td>4.0</td>
<td>0.6</td>
<td>0.26</td>
<td>4.0</td>
<td>-</td>
</tr>
<tr>
<td>tBUF</td>
<td>Bus free time between a STOP and START condition</td>
<td>4.7</td>
<td>1.3</td>
<td>0.5</td>
<td>4.7</td>
<td>-</td>
</tr>
<tr>
<td>tLOW</td>
<td>Low period of the SCL clock</td>
<td>4.7</td>
<td>1.3</td>
<td>0.5</td>
<td>4.7</td>
<td>-</td>
</tr>
<tr>
<td>tHIGH</td>
<td>Period of the SCL clock</td>
<td>4.0</td>
<td>0.6</td>
<td>0.26</td>
<td>4.0</td>
<td>50</td>
</tr>
<tr>
<td>tRise</td>
<td>Rise time of both SDA and SCL signals</td>
<td>-</td>
<td>1000</td>
<td>-</td>
<td>300</td>
<td>-</td>
</tr>
<tr>
<td>tFall</td>
<td>Fall time of both SDA and SCL signals</td>
<td>-</td>
<td>300</td>
<td>-</td>
<td>300</td>
<td>-</td>
</tr>
</tbody>
</table>

Note: SCLL is also used to generate the tBUF and tSU:STA timings, and SCLH is also used to generate the tHD:STA and tSU:STO timings.
Refer to Section 48.4.10 for examples of I2C_TIMINGR settings vs. i2c_ker_ck frequency.

Master communication initialization (address phase)

In order to initiate the communication, the user must program the following parameters for the addressed slave in the I2C_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configure to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

The user must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of tBUF.

In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.
**Note:** The START bit is reset by hardware when the slave address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs.

In 10-bit addressing mode, when the slave address first seven bits are NACKed by the slave, the master relaunches automatically the slave address transmission until ACK is received. In this case ADDRCF must be set if a NACK is received from the slave, to stop sending the slave address.

If the I2C is addressed as a slave (ADDR = 1) while the START bit is set, the I2C switches to slave mode and the START bit is cleared.

**Note:** The same procedure is applied for a repeated start condition. In this case BUSY = 1.

**Initialization of a master receiver addressing a 10-bit address slave**

- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set:
  (Re)Start + Slave address 10-bit header Write + Slave address second byte + REStart + Slave address 10-bit header Read

**Figure 636. 10-bit address read access with HEAD10R = 0**

---

**Figure 635. Master initialization flow**

 Initialization of a master receiver addressing a 10-bit address slave

- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set:
  (Re)Start + Slave address 10-bit header Write + Slave address second byte + REStart + Slave address 10-bit header Read

**Figure 636. 10-bit address read access with HEAD10R = 0**

---

**Figure 635. Master initialization flow**

 Initialization of a master receiver addressing a 10-bit address slave

- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set:
  (Re)Start + Slave address 10-bit header Write + Slave address second byte + REStart + Slave address 10-bit header Read

**Figure 636. 10-bit address read access with HEAD10R = 0**

---
If the master addresses a 10-bit address slave, transmits data to this slave and then reads data from the same slave, a master transmission flow must be done first. Then a repeated start is set with the 10-bit slave address configured with HEAD10R = 1. In this case the master sends this sequence: ReStart + Slave address 10-bit header Read.

**Figure 637. 10-bit address read access with HEAD10R = 1**

### Master transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit is set in the I2C_CR1 register. The flag is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD = 0 and NBYTES data have been transferred:
  - In automatic end mode (AUTOEND = 1), a STOP is automatically sent.
  - In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low in order to perform software actions:
    - A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.
    - A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. the NACKF flag is set in the I2C_ISR register, and an interrupt is generated if the NACKIE bit is set.
Figure 638. Transfer sequence flow for I2C master transmitter for $N \leq 255$ bytes
Figure 639. Transfer sequence flow for I2C master transmitter for N > 255 bytes

Master transmission

Master initialization

NBYTES = 0xFF; N=N-255
RELOAD = 1
Configure slave address
Set I2C_CR2.START

I2C_ISR.NACKF = 1?

No

Yes

End

I2C_ISR.TXIS = 1?

No

Yes

Write I2C_TXDR

NBYTES transmitted ?

No

Yes

Set I2C_CR2.START
with slave address
NBYTES = ...

I2C_ISR.TC = 1?

No

IF N< 256
NBYTES = N; N = 0; RELOAD = 0
AUTOEND = 0 for RESTART; 1 for STOP
ELSE
NBYTES = 0xFF; N = N-255
RELOAD = 1

End

No
Figure 640. Transfer bus diagrams for I2C master transmitter
(mandatory events only)

Example I2C master transmitter 2 bytes, automatic end mode (STOP)

INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2

Example I2C master transmitter 2 bytes, software end mode (RESTART)

INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START
Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C_CR1 register. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

• When RELOAD = 0 and NBYTES[7:0] data have been transferred:
  – In automatic end mode (AUTOEND = 1), a NACK and a STOP are automatically sent after the last received byte.
  – In software end mode (AUTOEND = 0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
Figure 641. Transfer sequence flow for I2C master receiver for $N \leq 255$ bytes

1. **Master initialization**
2. $NBYTES = N$
   - AUTOEND = 0 for RESTART, 1 for STOP
   - Configure slave address
   - Set I2C_CR2.START
3. I2C_ISR.RXNE = 1?
   - No
   - $NBYTES$ received?
     - No
     - Set I2C_CR2.START with slave address $NBYTES$...
     - End
   - Yes
     - Read I2C_RXDR
     - I2C_ISR.TC = 1?
       - No
       - End
       - Yes
         - Set I2C_CR2.START with slave address $NBYTES$...
Figure 642. Transfer sequence flow for I2C master receiver for \( N > 255 \) bytes

1. **Master initialization**
   - \( \text{NBYTES} = 0xFF; \text{N}=\text{N}-255 \)
   - \( \text{RELOAD} = 1 \)
   - Configure slave address
   - Set I2C_CR2.START

2. **I2C_ISR.RXNE = 1?**
   - Yes
     - Read I2C_RXDR

3. **NBYTES received?**
   - Yes
     - Set I2C_ISR.TC = 1?
     - No
       - IF \( N<256 \)
         - \( \text{NBYTES} = \text{N}; \text{N}=0; \text{RELOAD}=0 \)
         - \( \text{AUTOEND}=0 \) for RESTART, 1 for STOP
         - ELSE
           - \( \text{NBYTES} = 0xFF; \text{N}=\text{N}-255 \)
           - \( \text{RELOAD}=1 \)
     - Yes

4. **I2C_ISR.TCR = 1?**
   - No
     - IF \( \text{N} < 256 \)
       - \( \text{NBYTES} = \text{N}; \text{N}=0; \text{RELOAD}=0 \)
       - \( \text{AUTOEND}=0 \) for RESTART, 1 for STOP
       - ELSE
         - \( \text{NBYTES} = 0xFF; \text{N}=\text{N}-255 \)
         - \( \text{RELOAD}=1 \)

5. **End**
48.4.10 I2C_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I2C specification. To get more accurate configuration values, use the STM32CubeMX tool (I2C Configuration window).
48.4.11 SMBus specific features

This section is relevant only when the SMBus feature is supported (refer to Section 48.3).

### Table 505. Examples of timing settings for $f_{I^2CCLK}$ = 8 MHz

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 kHz</td>
<td>100 kHz</td>
<td>400 kHz</td>
</tr>
<tr>
<td>PRESC</td>
<td>0x1</td>
<td>0x1</td>
<td>0x0</td>
</tr>
<tr>
<td>SCLL</td>
<td>0xC7</td>
<td>0x13</td>
<td>0x9</td>
</tr>
<tr>
<td>$t_{SCLL}$</td>
<td>200 x 250 ns = 50 µs</td>
<td>20 x 250 ns = 5.0 µs</td>
<td>10 x 125 ns = 1250 ns</td>
</tr>
<tr>
<td>SCLH</td>
<td>0xC3</td>
<td>0xF</td>
<td>0x3</td>
</tr>
<tr>
<td>$t_{SCLH}$</td>
<td>196 x 250 ns = 49 µs</td>
<td>16 x 250 ns = 4.0 µs</td>
<td>4 x 125 ns = 500 ns</td>
</tr>
<tr>
<td>$t_{SCL}$ (1)</td>
<td>~100 µs(2)</td>
<td>~10 µs(2)</td>
<td>~2.5 µs(3)</td>
</tr>
<tr>
<td>SDADEL</td>
<td>0x2</td>
<td>0x2</td>
<td>0x1</td>
</tr>
<tr>
<td>$t_{SDADEL}$</td>
<td>2 x 250 ns = 500 ns</td>
<td>2 x 250 ns = 500 ns</td>
<td>1 x 125 ns = 125 ns</td>
</tr>
<tr>
<td>SCLDEL</td>
<td>0x4</td>
<td>0x4</td>
<td>0x3</td>
</tr>
<tr>
<td>$t_{SCLDEL}$</td>
<td>5 x 250 ns = 1250 ns</td>
<td>5 x 250 ns = 1250 ns</td>
<td>4 x 125 ns = 500 ns</td>
</tr>
</tbody>
</table>

1. $t_{SCL}$ is greater than $t_{SCLL} + t_{SCLH}$ due to SCL internal detection delay. Values provided for $t_{SCL}$ are examples only.
2. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x $t_{I^2CCLK} = 500$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 1000$ ns.
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x $t_{I^2CCLK} = 500$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 750$ ns.
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x $t_{I^2CCLK} = 500$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 655$ ns.

### Table 506. Examples of timing settings for $f_{I^2CCLK}$ = 16 MHz

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 kHz</td>
<td>100 kHz</td>
<td>400 kHz</td>
</tr>
<tr>
<td>PRESC</td>
<td>0x3</td>
<td>0x3</td>
<td>0x1</td>
</tr>
<tr>
<td>SCLL</td>
<td>0xC7</td>
<td>0x13</td>
<td>0x9</td>
</tr>
<tr>
<td>$t_{SCLL}$</td>
<td>200 x 250 ns = 50 µs</td>
<td>20 x 250 ns = 5.0 µs</td>
<td>10 x 125 ns = 1250 ns</td>
</tr>
<tr>
<td>SCLH</td>
<td>0xC3</td>
<td>0xF</td>
<td>0x3</td>
</tr>
<tr>
<td>$t_{SCLH}$</td>
<td>196 x 250 ns = 49 µs</td>
<td>16 x 250 ns = 4.0 µs</td>
<td>4 x 125 ns = 500 ns</td>
</tr>
<tr>
<td>$t_{SCL}$ (1)</td>
<td>~100 µs(2)</td>
<td>~10 µs(2)</td>
<td>~2.5 µs(3)</td>
</tr>
<tr>
<td>SDADEL</td>
<td>0x2</td>
<td>0x2</td>
<td>0x2</td>
</tr>
<tr>
<td>$t_{SDADEL}$</td>
<td>2 x 250 ns = 500 ns</td>
<td>2 x 250 ns = 500 ns</td>
<td>2 x 125 ns = 250 ns</td>
</tr>
<tr>
<td>SCLDEL</td>
<td>0x4</td>
<td>0x4</td>
<td>0x3</td>
</tr>
<tr>
<td>$t_{SCLDEL}$</td>
<td>5 x 250 ns = 1250 ns</td>
<td>5 x 250 ns = 1250 ns</td>
<td>4 x 125 ns = 500 ns</td>
</tr>
</tbody>
</table>

1. $t_{SCL}$ is greater than $t_{SCLL} + t_{SCLH}$ due to SCL internal detection delay. Values provided for $t_{SCL}$ are examples only.
2. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x $t_{I^2CCLK} = 250$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 1000$ ns.
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x $t_{I^2CCLK} = 250$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 750$ ns.
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4 x $t_{I^2CCLK} = 250$ ns. Example with $t_{SYNC1} + t_{SYNC2} = 500$ ns.
Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I²C principles of operation. The SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBus specification (http://smbus.org).

The system management bus specification refers to three types of devices

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks, and terminates the transfer.
- A host is a specialized master that provides the main interface to the system’s CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as master or slave device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. A device can use any or all of them to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write, and Block Write-Block Read Process Call. These protocols must be implemented by the user software.

For more details on these protocols, refer to SMBus specification (http://smbus.org).

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. To provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2C_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of the SMBus address resolution protocol, refer to SMBus specification (http://smbus.org).

Received command and data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting SBC bit in I2C_CR1 register. Refer to Slave byte control mode for more details.

Host notify protocol

This peripheral supports the host notify protocol by setting the SMBHEN bit in the I2C_CR1 register. In this case the host acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.
SMBus alert

The SMBus ALERT optional signal is supported. A slave-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device(s) which pulled SMBALERT# low acknowledges the alert response address.

When configured as a slave device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN = 1), the ALERT flag is set in the I2C_ISR register when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. When ALERTEN = 0, the ALERT line is considered high even if the external SMBALERT pin is low.

*If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN = 0.*

Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match with the hardware calculated PEC.

Timeouts

This peripheral embeds hardware timers in order to be compliant with the three timeouts defined in SMBus specification.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{TIMEOUT}$</td>
<td>Detect clock low timeout</td>
<td>25</td>
<td>35</td>
</tr>
<tr>
<td>$t_{LOW:SEXT}^{(1)}$</td>
<td>Cumulative clock low extend time (slave device)</td>
<td>-</td>
<td>25</td>
</tr>
<tr>
<td>$t_{LOW:MEXT}^{(2)}$</td>
<td>Cumulative clock low extend time (master device)</td>
<td>-</td>
<td>10</td>
</tr>
</tbody>
</table>

1. $t_{LOW:SEXT}$ is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another slave device or the master also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.

2. $t_{LOW:MEXT}$ is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a slave device or another master also extends the clock, causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.
Bus idle detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for \( t_{\text{IDLE}} > t_{\text{HIGH,MAX}} \) (refer to I2C timings).

This timing parameter covers the condition where a master has been dynamically added to the bus, and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

48.4.12 SMBus initialization

This section is relevant only when SMBus feature is supported (see Section 48.3).

In addition to I2C initialization, some other specific initialization must be done to perform SMBus communication.

Received command and data acknowledge control (slave mode)

A SMBus receiver must be able to NACK each received command or data. To allow ACK control in slave mode, the Slave byte control mode must be enabled by setting the SBC bit in the I2C_CR1 register. Refer to Slave byte control mode for more details.

Specific address (slave mode)

The specific SMBus addresses must be enabled if needed. Refer to Bus idle detection for more details.

- The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit in the I2C_CR1 register.
- The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit in the I2C_CR1 register.
- The alert response address (0b0001100) is enabled by setting the ALERTEN bit in the I2C_CR1 register.
Packet error checking

PEC calculation is enabled by setting the PECEN bit in the I2C_CR1 register. Then the PEC transfer is managed with the help of the hardware byte counter NBYTES[7:0] in the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in slave mode. The PEC is transferred after NBYTES - 1 data have been transferred when the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C is enabled.

<table>
<thead>
<tr>
<th>Table 508. SMBus with PEC configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mode</strong></td>
</tr>
<tr>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>Master Tx/Rx NBYTES + PEC+ STOP</td>
</tr>
<tr>
<td>Master Tx/Rx NBYTES + PEC + ReSTART</td>
</tr>
<tr>
<td>Slave Tx/Rx with PEC</td>
</tr>
</tbody>
</table>

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits in the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

- **t_TIMEOUT check**
  
  To enable the t_TIMEOUT check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value, to check the t_TIMEOUT parameter. The TIDLE bit must be configured to 0 to detect the SCL low level timeout.

  Then the timer is enabled by setting the TIMOUTEN in the I2C_TIMEOUTR register.

  If SCL is tied low for a time greater than (TIMEOUTA + 1) x 2048 x tI2CCLK, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to Table 509.

Caution: Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMOUTEN bit is set.

- **t_LOW:SEXT and t_LOW:MEXT check**

  Depending on if the peripheral is configured as a master or as a slave, the 12-bit TIMEOUTB timer must be configured to check t_LOW:SEXT for a slave, and t_LOW:MEXT for a master. As the standard specifies only a maximum, the user can choose the same value for both. The timer is then enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

  If the SMBus peripheral performs a cumulative SCL stretch for a time greater than (TIMEOUTB + 1) x 2048 x tI2CCLK, and in the timeout interval described in *Bus idle detection* section, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to Table 510

Caution: Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.
Bus idle detection

To enable the \( t_{\text{idle}} \) check, the 12-bit \( \text{TIMEOUTA}[11:0] \) field must be programmed with the timer reload value, to obtain the \( t_{\text{idle}} \) parameter. The TIDLE bit must be configured to ‘1’ to detect both SCL and SDA high level timeout. The timer is then enabled by setting the TIMOUTEN bit in the I2C\_TIMEOUTR register.

If both the SCL and SDA lines remain high for a time greater than \( (\text{TIMEOUTA} + 1) \times 4 \times \text{I2CCLK} \), the TIMEOUT flag is set in the I2C\_ISR register.

Refer to Table 511.

Caution: Changing \( \text{TIMEOUTA} \) and TIDLE configuration is not allowed when \( \text{TIMEOUTEN} \) is set.

48.4.13 SMBus: I2C\_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to Section 48.3.

- Configuring the maximum duration of \( t_{\text{TIMEOUT}} \) to 25 ms:

  Table 509. Examples of \( \text{TIMEOUTA} \) settings (max \( t_{\text{TIMEOUT}} = 25 \text{ ms} \))

<table>
<thead>
<tr>
<th>( f\text{I2CCLK} )</th>
<th>( \text{TIMEOUTA}[11:0] ) bits</th>
<th>TIDLE bit</th>
<th>TIMEOUTEN bit</th>
<th>( t_{\text{TIMEOUT}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 MHz</td>
<td>0x61</td>
<td>0</td>
<td>1</td>
<td>( 98 \times 2048 \times 125 \text{ ns} = 25 \text{ ms} )</td>
</tr>
<tr>
<td>16 MHz</td>
<td>0xC3</td>
<td>0</td>
<td>1</td>
<td>( 196 \times 2048 \times 62.5 \text{ ns} = 25 \text{ ms} )</td>
</tr>
</tbody>
</table>

- Configuring the maximum duration of \( t_{\text{LOW:SEXT}} \) and \( t_{\text{LOW:MEXT}} \) to 8 ms:

  Table 510. Examples of \( \text{TIMEOUTB} \) settings

<table>
<thead>
<tr>
<th>( f\text{I2CCLK} )</th>
<th>( \text{TIMEOUTB}[11:0] ) bits</th>
<th>TEXTEN bit</th>
<th>( t_{\text{LOW:EXT}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 MHz</td>
<td>0x1F</td>
<td>1</td>
<td>( 32 \times 2048 \times 125 \text{ ns} = 8 \text{ ms} )</td>
</tr>
<tr>
<td>16 MHz</td>
<td>0x3F</td>
<td>1</td>
<td>( 64 \times 2048 \times 62.5 \text{ ns} = 8 \text{ ms} )</td>
</tr>
</tbody>
</table>

- Configuring the maximum duration of \( t_{\text{IDLE}} \) to 50 µs

  Table 511. Examples of \( \text{TIMEOUTA} \) settings (max \( t_{\text{IDLE}} = 50 \text{ µs} \))

<table>
<thead>
<tr>
<th>( f\text{I2CCLK} )</th>
<th>( \text{TIMEOUTA}[11:0] ) bits</th>
<th>TIDLE bit</th>
<th>TIMEOUTEN bit</th>
<th>( t_{\text{IDLE}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 MHz</td>
<td>0x63</td>
<td>1</td>
<td>1</td>
<td>( 100 \times 4 \times 125 \text{ ns} = 50 \text{ µs} )</td>
</tr>
<tr>
<td>16 MHz</td>
<td>0xC7</td>
<td>1</td>
<td>1</td>
<td>( 200 \times 4 \times 62.5 \text{ ns} = 50 \text{ µs} )</td>
</tr>
</tbody>
</table>

48.4.14 SMBus slave mode

This section is relevant only when the SMBus feature is supported (refer to Section 48.3).

In addition to I2C slave transfer management (refer to Section 48.4.8), additional software flows are provided to support the SMBus.

SMBus slave transmitter

When the IP is used in SMBus, SBC must be programmed to 1 to enable the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit
is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case the total number of TXIS interrupts is NBYTES - 1, and the content of the I2C_PECR register is automatically transmitted if the master requests an extra byte after the NBYTES - 1 data transfer.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 645. Transfer sequence flow for SMBus slave transmitter N bytes + PEC
When the I2C is used in SMBus mode, SBC must be programmed to 1 to allow the PEC checking at the end of the programmed number of data bytes. In order to allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to Slave byte control mode for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after NBYTES - 1 data have been received, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

If no ACK software control is needed, the user can program PECBYTE = 1 and, in the same write operation, program NBYTES with the number of bytes to be received in a continuous flow. After NBYTES - 1 are received, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.
Figure 647. Transfer sequence flow for SMBus slave receiver N bytes + PEC

SMBus slave reception

Slave initialization

No

I2C_ISR.ADDR = 1?

Yes

Read ADDCODE and DIR in I2C_ISR
I2C_CR2.NBYTES = 1, RELOAD =1
PECBYTE=1
Set I2C_ICR.ADDRCF

I2C_ISR.RXNE =1?
I2C_ISR.TCR = 1?

No

Yes

SCL stretched

Read I2C_RXDR.RXDATA
Program I2C_CR2.NACK = 0
I2C_CR2.NBYTES = 1
N = N - 1

N = 1?

No

Yes

Read I2C_RXDR.RXDATA
Program RELOAD = 0
NACK = 0 and NBYTES = 1

No

I2C_ISR.RXNE =1?

Yes

Read I2C_RXDR.RXDATA

End
This section is relevant only when the SMBus feature is supported (refer to Section 48.3). In addition to I2C master transfer management (refer to Section 48.4.9), additional software flows are provided to support the SMBus.

**SMBus master transmitter**

When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTES - 1. So if the PECBYTE bit is set when NBYTES = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus master wants to send a STOP condition after the PEC, automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus master wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND = 0). In this case, once NBYTES - 1 have been
transmitted, the I2C_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 649. Bus transfer diagrams for SMBus master transmitter

Example SMBus master transmitter 2 bytes + PEC, automatic end mode (STOP)

Example SMBus master transmitter 2 bytes + PEC, software end mode (RESTART)
**SMBus master receiver**

When the SMBus master wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES - 1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES - 1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.
Figure 650. Bus transfer diagrams for SMBus master receiver

Example SMBus master receiver 2 bytes + PEC, automatic end mode (STOP)

INIT: program Slave address, program NBYTES = 3, AUTOEND=1, set PECBYTE, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2
EV3: RXNE ISR: rd PEC

Example SMBus master receiver 2 bytes + PEC, software end mode (RESTART)

INIT: program Slave address, program NBYTES = 3, AUTOEND=0, set PECBYTE, set START
EV1: RXNE ISR: rd data1
EV2: RXNE ISR: rd data2
EV3: RXNE ISR: read PEC
EV4: TC ISR: program Slave address, program NBYTES = N, set START
48.4.15 Wake-up from Stop mode on address match

This section is relevant only when wake-up from Stop mode feature is supported (refer to Section 48.3).

The I2C is able to wake-up the MCU from Stop mode (APB clock is off), when it is addressed. All addressing modes are supported.

Wake-up from Stop mode is enabled by setting the WUPEN bit in the I2C_CR1 register. The HSI and CSI only oscillator must be selected as the clock source for I2CCLK in order to allow wake-up from Stop mode.

During Stop mode, the HSI and CSI only is switched off. When a START is detected, the I2C interface switches the HSI and CSI only on, and stretches SCL low until HSI and CSI only is woken up.

HSI and CSI only is then used for the address reception.

In case of an address match, the I2C stretches SCL low during MCU wake-up time. The stretch is released when ADDR flag is cleared by software, and the transfer goes on normally.

If the address does not match, the HSI and CSI only is switched off again and the MCU is not woken up.

Note: If the I2C clock is the system clock, or if WUPEN = 0, the HSI and CSI only is not switched on after a START is received.

Only an ADDR interrupt can wake-up the MCU. Therefore do not enter Stop mode when the I2C is performing a transfer as a master, or as an addressed slave after the ADDR flag is set. This can be managed by clearing SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

Caution: The digital filter is not compatible with the wake-up from Stop mode feature. If the DNF bit is not equal to 0, setting the WUPEN bit has no effect.

Caution: This feature is available only when the I2C clock source is the HSI and CSI only oscillator.

Caution: Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

Caution: If wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

48.4.16 Error conditions

The following errors are the conditions that can cause a communication fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of nine SCL clock pulses. A START or a STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as master or addressed slave (i.e not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C enters address recognition state like for a correct START condition.
When a bus error is detected, the BERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

**Arbitration lost (ARLO)**

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

- In master mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the master switches automatically to slave mode.
- In slave mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

**Overrun/underrun error (OVR)**

An overrun or underrun error is detected in slave mode when NOSTRETCH = 1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF = 1 and the first data byte must be sent. The content of the I2C_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

**Packet error checking error (PECERR)**

This section is relevant only when the SMBus feature is supported (refer to Section 48.3).

A PEC error is detected when the received PEC byte does not match with the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.
Timeout error (TIMEOUT)

This section is relevant only when the SMBus feature is supported (refer to Section 48.3).

A timeout error occurs for any of these conditions:

- $\text{TIDLE} = 0$ and $\text{SCL}$ remained low for the time defined in the TIMEOUTA[11:0] bits: this is used to detect an SMBus timeout.
- $\text{TIDLE} = 1$ and both $\text{SDA}$ and $\text{SCL}$ remained high for the time defined in the TIMEOUTA[11:0] bits: this is used to detect a bus idle condition.
- Master cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus $t_{\text{LOW:MEXT}}$ parameter).
- Slave cumulative clock low extend time reached the time defined in TIMEOUTB[11:0] bits (SMBus $t_{\text{LOW:SEXT}}$ parameter).

When a timeout violation is detected in master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Alert (ALERT)

This section is relevant only when the SMBus feature is supported (refer to Section 48.3).

The ALERT flag is set when the I2C interface is configured as a Host ($\text{SMBHEN} = 1$), the alert pin detection is enabled ($\text{ALERTEN} = 1$) and a falling edge is detected on the $\text{SMBA}$ pin. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

48.4.17 DMA requests

Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit in the I2C_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see Section 11: Direct memory access controller (DMA) on page 359) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to Master transmitter.
- In slave mode:
  - With $\text{NOSTRETCH} = 0$, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With $\text{NOSTRETCH} = 1$, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to SMBus slave transmitter and SMBus master transmitter.
**Note:** If DMA is used for transmission, the TXIE bit does not need to be enabled.

**Reception using DMA**

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit in the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured using the DMA peripheral (refer to Section 11: Direct memory access controller (DMA) on page 359) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.
- In slave mode with NOSTRETCH = 0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.
- If SMBus is supported (see Section 48.3) the PEC transfer is managed with the NBYTES counter. Refer to SMBus slave receiver and SMBus master receiver.

**Note:** If DMA is used for reception, the RXIE bit does not need to be enabled.

### 48.4.18 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG_I2Cx_ configuration bits in the DBG module.

### 48.5 I2C low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. I2C interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop(1)</td>
<td>The I2C registers content is kept.</td>
</tr>
<tr>
<td></td>
<td>– WUPEN = 1 and I2C is clocked by an internal oscillator (HSI and CSI only); the address recognition is functional. The I2C address match condition causes the device to exit the Stop mode.</td>
</tr>
<tr>
<td></td>
<td>– WUPEN = 0: the I2C must be disabled before entering Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The I2C peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

1. Refer to Section 48.3 for information about the Stop modes supported by each instance. If wake-up from a specific Stop mode is not supported, the instance must be disabled before entering this Stop mode.
### 48.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

**Table 513. I2C interrupt requests**

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit Sleep mode</th>
<th>Exit Stop modes</th>
<th>Exit Standby modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C_EV</td>
<td>Receive buffer not empty</td>
<td>RXNE</td>
<td>RXIE</td>
<td>Read I2C_RXDR register</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transmit buffer interrupt status</td>
<td>TXIS</td>
<td>TXIE</td>
<td>Write I2C_TXDR register</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Stop detection interrupt flag</td>
<td>STOPF</td>
<td>STOPIE</td>
<td>Write STOPCF = 1</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transfer complete reload</td>
<td>TCR</td>
<td>TCIE</td>
<td>Write I2C.CR2 with NBYTES[7:0] ≠ 0</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transfer complete</td>
<td>TC</td>
<td></td>
<td>Write START = 1 or STOP = 1</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Address matched</td>
<td>ADDR</td>
<td>ADDRIE</td>
<td>Write ADDRCF = 1</td>
<td>Yes(1)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>NACK reception</td>
<td>NACKF</td>
<td>NACKIE</td>
<td>Write NACKCF = 1</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>I2C_ER</td>
<td>Bus error</td>
<td>BERR</td>
<td></td>
<td>Write BERRCF = 1</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Arbitration loss</td>
<td>ARLO</td>
<td></td>
<td>Write ARLOCF = 1</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Overrun/Underrun</td>
<td>OVR</td>
<td></td>
<td>Write OVRCF = 1</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>PEC error</td>
<td>PECERR</td>
<td>ERRIE</td>
<td>Write PECERRCF = 1</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Timeout/tLOW error</td>
<td>TIMEOUT</td>
<td></td>
<td>Write TIMEOUTCF = 1</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>SMBus alert</td>
<td>ALERT</td>
<td></td>
<td>Write ALERTCF = 1</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

1. The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the Wake-up from Stop mode feature. Refer to Section 48.3.
48.7 I2C registers

Refer to Section 1.2 on page 101 for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

48.7.1 I2C control register 1 (I2C_CR1)

Address offset: 0x00
Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RXDMAEN</td>
<td>TXDMAEN</td>
<td>Res</td>
<td>ANF</td>
<td>OFF</td>
<td>DNF[3:0]</td>
<td>ERRIE</td>
<td>TCIE</td>
<td>STOPIE</td>
<td>NACKIE</td>
<td>ADDRIE</td>
<td>RXIE</td>
<td>TXIE</td>
<td>PE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>
```

Bit 31 STOPFACL: STOP detection flag (STOPF) automatic clear
0: STOPF flag is set by hardware, cleared by software by setting STOPCF bit.
1: STOPF flag remains cleared by hardware. This mode can be used in NOSTRETCH slave mode, to avoid the overrun error if the STOPF flag is not cleared before next data transmission. This allows a slave data management by DMA only, without any interrupt from peripheral.

Bit 30 ADDRACL: Address match flag (ADDR) automatic clear
0: ADDR flag is set by hardware, cleared by software by setting ADDRCF bit.
1: ADDR flag remains cleared by hardware. This mode can be used in slave mode, to avoid the ADDR clock stretching if the I2C enables only one slave address. This allows a slave data management by DMA only, without any interrupt from peripheral.

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 FMP: Fast-mode Plus 20 mA drive enable
0: 20 mA I/O drive disabled
1: 20 mA I/O drive enabled

Bit 23 PECEN: PEC enable
0: PEC calculation disabled
1: PEC calculation enabled

Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 48.3.
### Bit 22  **ALERTEN**: SMBus alert enable

- **0**: The SMBus alert pin (SMBA) is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the Alert Response Address header is disabled (0001100x followed by NACK).
- **1**: The SMBus alert pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the Alert Response Address header is enabled (0001100x followed by ACK).

**Note**: When **ALERTEN** = 0, the SMBA pin can be used as a standard GPIO.

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 48.3.*

### Bit 21  **SMBDEN**: SMBus device default address enable

- **0**: Device default address disabled. Address 0b1100001x is NACKed.
- **1**: Device default address enabled. Address 0b1100001x is ACKed.

**Note**: If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 48.3.

### Bit 20  **SMBHEN**: SMBus host address enable

- **0**: Host address disabled. Address 0b0001000x is NACKed.
- **1**: Host address enabled. Address 0b0001000x is ACKed.

**Note**: If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 48.3.

### Bit 19  **GCEN**: General call enable

- **0**: General call disabled. Address 0b00000000 is NACKed.
- **1**: General call enabled. Address 0b00000000 is ACKed.

### Bit 18  **WUPEN**: Wake-up from Stop mode enable

- **0**: Wake-up from Stop mode disable.
- **1**: Wake-up from Stop mode enable.

**Note**: If the wake-up from Stop mode feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 48.3.

**Note**: **WUPEN** can be set only when DNF = ‘0000’

### Bit 17  **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in slave mode. It must be kept cleared in master mode.

- **0**: Clock stretching enabled
- **1**: Clock stretching disabled

**Note**: This bit can be programmed only when the I2C is disabled (PE = 0).

### Bit 16  **SBC**: Slave byte control

- **0**: Slave byte control disabled
- **1**: Slave byte control enabled

### Bit 15  **RXDMAEN**: DMA reception requests enable

- **0**: DMA mode disabled for reception
- **1**: DMA mode enabled for reception

### Bit 14  **TXDMAEN**: DMA transmission requests enable

- **0**: DMA mode disabled for transmission
- **1**: DMA mode enabled for transmission

### Bit 13  Reserved, must be kept at reset value.
Bit 12 ANFOFF: Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled

Note: This bit can be programmed only when the I2C is disabled (PE = 0).

Bits 11:8 DNF[3:0]: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * tI2CCLK
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to 1 tI2CCLK
...
1111: digital filter enabled and filtering capability up to 15 tI2CCLK

Note: If the analog filter is enabled, the digital filter is added to it.
This filter can be programmed only when the I2C is disabled (PE = 0).

Bit 7 ERRIE: Error interrupts enable
0: Error detection interrupts disabled
1: Error detection interrupts enabled

Note: Any of these errors generate an interrupt:
Arbitration loss (ARLO)
Bus error detection (BERR)
Overrun/Underrun (OVR)
Timeout detection (TIMEOUT)
PEC error detection (PECERR)
Alert pin event detection (ALERT)

Bit 6 TCIE: Transfer complete interrupt enable
0: Transfer complete interrupt disabled
1: Transfer complete interrupt enabled

Note: Any of these events generate an interrupt:
Transfer complete (TC)
Transfer complete reload (TCR)

Bit 5 STOPIE: Stop detection interrupt enable
0: Stop detection (STOPF) interrupt disabled
1: Stop detection (STOPF) interrupt enabled

Bit 4 NACKIE: Not acknowledge received interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 ADDRIE: Address match interrupt enable (slave only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 RXIE: RX interrupt enable
0: Receive (RXNE) interrupts disabled
1: Receive (RXNE) interrupts enabled

Bit 1 TXIE: TX interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled
Bit 0 **PE**: Peripheral enable
- 0: Peripheral disable
- 1: Peripheral enable

*Note:* When PE = 0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least three APB clock cycles.

### 48.7.2 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>NACK</td>
<td>STOP</td>
<td>START</td>
<td>HEAD10R</td>
<td>ADDD10</td>
<td>RD_WRN</td>
<td>SADD[9:0]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs</td>
<td>rs</td>
</tr>
</tbody>
</table>

**Bits 31:27** Reserved, must be kept at reset value.

**Bit 26** **PECBYTE**: Packet error checking byte

- This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.
- 0: No PEC transfer.
- 1: PEC transmission/reception is requested

*Note:* Writing 0 to this bit has no effect.

*This bit has no effect when RELOAD is set.*
*This bit has no effect is slave mode when SBC = 0.*
*If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0.*
*Refer to Section 48.3.*

**Bit 25** **AUTOEND**: Automatic end mode (master mode)

- This bit is set and cleared by software.
- 0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.
- 1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

*Note:* This bit has no effect in slave mode or when the RELOAD bit is set.

**Bit 24** **RELOAD**: NBYTES reload mode

- This bit is set and cleared by software.
- 0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).
- 1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.
Bits 23:16 **NBYES[7:0]**: Number of bytes
The number of bytes to be transmitted/received is programmed there. This field is don’t care in slave mode with SBC = 0.

*Note: Changing these bits when the START bit is set is not allowed.*

Bit 15 **NACK**: NACK generation (slave mode)
The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.
0: an ACK is sent after current received byte.
1: a NACK is sent after current received byte.

*Note: Writing 0 to this bit has no effect.*

This bit is used in slave mode only: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.
When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated, whatever the NACK bit value.
When hardware PEC checking is enabled (PECBYTE = 1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: Stop generation (master mode)
The bit is set by software, cleared by hardware when a STOP condition is detected, or when PE = 0.

*In master mode:*
0: No Stop generation.
1: Stop generation after current byte transfer.

*Note: Writing 0 to this bit has no effect.*

Bit 13 **START**: Start generation
This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by an address matched in slave mode, by a timeout error detection, or when PE = 0.
0: No Start generation.
1: Restart/Start generation:
If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated start condition when RELOAD = 0, after the end of the NBYES transfer.
Otherwise setting this bit generates a START condition once the bus is free.

*Note: Writing 0 to this bit has no effect.*

The START bit can be set even if the bus is BUSY or I2C is in slave mode.
The bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (master receiver mode)
0: The master sends the complete 10-bit slave address read sequence: Start + 2 bytes 10-bit address in write direction + Restart + first seven bits of the 10-bit address in read direction.
1: The master only sends the first seven bits of the 10-bit address, followed by Read direction.

*Note: Changing this bit when the START bit is set is not allowed.*

Bit 11 **ADD10**: 10-bit addressing mode (master mode)
0: The master operates in 7-bit addressing mode,
1: The master operates in 10-bit addressing mode

*Note: Changing this bit when the START bit is set is not allowed.*
48.7.3 I2C own address 1 register (I2C_OAR1)

Address offset: 0x08
Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

| Bit 31:16 | Reserved, must be kept at reset value. |
| Bit 15 | OA1EN: Own address 1 enable |
| 0: Own address 1 disabled. The received slave address OA1 is NACKed. |
| 1: Own address 1 enabled. The received slave address OA1 is ACKed. |
| Bit 14:11 | Reserved, must be kept at reset value. |
| Bit 10 | OA1MODE: Own address 1 10-bit mode |
| 0: Own address 1 is a 7-bit address. |
| 1: Own address 1 is a 10-bit address. |
| Note: This bit can be written only when OA1EN = 0. |
| Bits 9:0 | OA1[9:0]: Interface own slave address |
| 7-bit addressing mode: OA1[7:1] contains the 7-bit own slave address. The bits OA1[9], OA1[8] and OA1[0] are don’t care. |
| 10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address. |
| Note: These bits can be written only when OA1EN = 0. |
48.7.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C
Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2x i2c_pclk + 6 x i2c_ker_ck.

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0C</td>
<td>I2C_OAR2</td>
<td></td>
<td>RW</td>
</tr>
<tr>
<td>31-16</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>OA2EN</td>
<td></td>
<td>RW</td>
</tr>
<tr>
<td>14-11</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-8</td>
<td>OA2MSK[2:0]</td>
<td></td>
<td>RW</td>
</tr>
<tr>
<td>7-1</td>
<td>OA2[7:1]</td>
<td></td>
<td>RW</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own address 2 enable
0: Own address 2 disabled. The received slave address OA2 is NACKed.
1: Own address 2 enabled. The received slave address OA2 is ACKed.

Bits 14:11  Reserved, must be kept at reset value.

Bits 10:8  **OA2MSK[2:0]**: Own address 2 masks
000: No mask
001: OA2[1] is masked and don’t care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don’t care. Only OA2[7:3] are compared.
100: OA2[4:1] are masked and don’t care. Only OA2[7:5] are compared.
111: OA2[7:1] are masked and don’t care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

*Note:* These bits can be written only when OA2EN = 0.
As soon as OA2MSK is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if the comparison matches.

Bits 7:1  **OA2[7:1]**: Interface address

7-bit addressing mode: 7-bit address

*Note:* These bits can be written only when OA2EN = 0.

Bit 0  Reserved, must be kept at reset value.
48.7.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10
Reset value: 0x0000 0000
Access: no wait states

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw   rw   rw   rw   rw   rw   rw   rw   rw   rw   rw   rw</td>
</tr>
<tr>
<td>Bits 31:28 PRESC[3:0]: Timing prescaler</td>
</tr>
</tbody>
</table>
| This field is used to prescale i2c_ker_ck in order to generate the clock period t<sub>PRESC</sub> used for data setup and hold counters (refer to I2C timings) and for SCL high and low level counters (refer to I2C master initialization).
| t<sub>PRESC</sub> = (PRESC + 1) x t<sub>I2CCLK</sub> |
| Bits 27:24 Reserved, must be kept at reset value. |
| Bits 23:20 SCLDEL[3:0]: Data setup time |
| This field is used to generate a delay t<sub>SCLDEL</sub> between SDA edge and SCL rising edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t<sub>SCLDEL</sub>.
| t<sub>SCLDEL</sub> = (SCLDEL + 1) x t<sub>PRESC</sub> |
| Note: t<sub>SCLDEL</sub> is used to generate t<sub>SU:DAT</sub> timing. |
| Bits 19:16 SDADEL[3:0]: Data hold time |
| This field is used to generate the delay t<sub>SDADEL</sub> between SCL falling edge and SDA edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t<sub>SDADEL</sub>.
| t<sub>SDADEL</sub> = SDADEL x t<sub>PRESC</sub> |
| Note: SDADEL is used to generate t<sub>HD:DAT</sub> timing. |
| Bits 15:8 SCLH[7:0]: SCL high period (master mode) |
| This field is used to generate the SCL high period in master mode.
| t<sub>SCLH</sub> = (SCLH + 1) x t<sub>PRESC</sub> |
| Note: SCLH is also used to generate t<sub>SU:STO</sub> and t<sub>HD:STA</sub> timing. |
| Bits 7:0 SCLL[7:0]: SCL low period (master mode) |
| This field is used to generate the SCL low period in master mode.
| t<sub>SCLL</sub> = (SCLL + 1) x t<sub>PRESC</sub> |
| Note: SCLL is also used to generate t<sub>BUF</sub> and t<sub>SU:STA</sub> timings. |

Note: This register must be configured when the I2C is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.
I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14
Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

<table>
<thead>
<tr>
<th>TEXTEN</th>
<th>TIMOUTEN</th>
<th>TIDLE</th>
<th>TIMEOUTB[11:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
</tr>
<tr>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
</tr>
<tr>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>

Bit 31 **TEXTEN**: Extended clock timeout enable
0: Extended clock timeout detection is disabled
1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than \( t_{LOW:EXT} \) is done by the I2C interface, a timeout error is detected (TIMEOUT = 1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B
This field is used to configure the cumulative clock extension timeout:
- In master mode, the master cumulative clock low extend time \( t_{LOW:MEXT} \) is detected
- In slave mode, the slave cumulative clock low extend time \( t_{LOW:SEXT} \) is detected
\[ t_{LOW:EXT} = (\text{TIMEOUTB} + \text{TIDLE = 0}) \times 2048 \times \text{i2c_clk} \]
Note: These bits can be written only when TEXTEN = 0.

Bit 15 **TIMOUTEN**: Clock timeout enable
0: SCL timeout detection is disabled
1: SCL timeout detection is enabled: when SCL is low for more than \( t_{TIMEOUT} \) (TIDLE = 0) or high for more than \( t_{IDLE} \) (TIDLE = 1), a timeout error is detected (TIMEOUT = 1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection
0: TIMEOUTA is used to detect SCL low timeout
1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)
Note: This bit can be written only when TIMOUTEN = 0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus Timeout A
This field is used to configure:
The SCL low timeout condition \( t_{TIMEOUT} \) when TIDLE = 0
\[ t_{TIMEOUT} = (\text{TIMEOUTA} + 1) \times 2048 \times \text{i2c_clk} \]
The bus idle condition (both SCL and SDA high) when TIDLE = 1
\[ t_{IDLE} = (\text{TIMEOUTA} + 1) \times 4 \times \text{i2c_clk} \]
Note: These bits can be written only when TIMOUTEN = 0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to “0x00000000”. Refer to Section 48.3.
48.7.7 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18
Reset value: 0x0000 0001
Access: no wait states

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>BUSY</td>
<td>Res.</td>
<td>ALERT</td>
<td>TIME OUT</td>
<td>PEC</td>
<td>ERR</td>
<td>OVR</td>
<td>ARLO</td>
<td>BERR</td>
<td>TCR</td>
<td>TC</td>
<td>STOPF</td>
<td>NACKF</td>
<td>ADDR</td>
<td>RXNE</td>
<td>TXIS</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>rs</td>
<td>rs</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 ADDCODE[6:0]: Address match code (slave mode)
These bits are updated with the received address when an address match event occurs (ADDR = 1).
In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the two MSBs of the address.

Bit 16 **DIR**: Transfer direction (slave mode)
This flag is updated when an address match event occurs (ADDR = 1).
0: Write transfer, slave enters receiver mode.
1: Read transfer, slave enters transmitter mode.

Bit 15 **BUSY**: Bus busy
This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected, or when PE = 0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert
This flag is set by hardware when SMBHEN = 1 (SMBus host configuration), ALERTEN = 1 and an SMBALERT event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

*Note:* This bit is cleared by hardware when PE = 0.
If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 48.3.

Bit 12 **TIMEOUT**: Timeout or t\text{LOW} detection flag
This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

*Note:* This bit is cleared by hardware when PE = 0.
If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 48.3.
Bit 11 **PECERR**: PEC Error in reception
This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 48.3.*

Bit 10 **OVR**: Overrun/Underrun (slave mode)
This flag is set by hardware in slave mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 9 **ARLO**: Arbitration lost
This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 8 **BERR**: Bus error
This flag is set by hardware when a misplaced Start or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in slave mode. It is cleared by software by setting **BERRCF bit**.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 7 **TCR**: Transfer Complete Reload
This flag is set by hardware when RELOAD = 1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

*Note: This bit is cleared by hardware when PE = 0.*

**This flag is only for master mode, or for slave mode when the SBC bit is set.**

Bit 6 **TC**: Transfer Complete (master mode)
This flag is set by hardware when RELOAD = 0, AUTOEND = 0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 5 **STOPF**: Stop detection flag
This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:
- either as a master, provided that the STOP condition is generated by the peripheral.
- or as a slave, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 4 **NACKF**: Not Acknowledge received flag
This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 3 **ADDR**: Address matched (slave mode)
This bit is set by hardware as soon as the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting **ADDRCF bit**.

*Note: This bit is cleared by hardware when PE = 0.*
Bit 2 **RXNE**: Receive data register not empty (receivers)
This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 1 **TXIS**: Transmit interrupt status (transmitters)
This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.
This bit can be written to 1 by software only when NOSTRETCH = 1, to generate a TXIS event (interrupt if TXIE = 1 or DMA request if TXDMAEN = 1).

*Note: This bit is cleared by hardware when PE = 0.*

Bit 0 **TXE**: Transmit data register empty (transmitters)
This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.
This bit can be written to 1 by software in order to flush the transmit data register I2C_TXDR.

*Note: This bit is set by hardware when PE = 0.*

### 48.7.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C
Reset value: 0x0000 0000
Access: no wait states

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear
Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 48.3.*

Bit 12 **TIMOUTCF**: Timeout detection flag clear
Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 48.3.*

Bit 11 **PECCF**: PEC Error flag clear
Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to 0. Refer to Section 48.3.*

Bit 10 **OVRCF**: Overrun/Underrun flag clear
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.
Bit 9 **ARLOCF**: Arbitration lost flag clear
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not Acknowledge flag clear
Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

### 48.7.9 I2C PEC register (I2C_PECR)

Address offset: 0x20
Reset value: 0x0000 0000
Access: no wait states

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PEC[7:0]**: Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE = 0.

**Note:** If the SMBus feature is not supported, this register is reserved and forced by hardware to “0x00000000”. Refer to Section 48.3.
48.7.10  I2C receive data register (I2C_RXDR)

Address offset: 0x24
Reset value: 0x0000 0000
Access: no wait states

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0  \textbf{RXDATA}[7:0]: 8-bit receive data
Data byte received from the I2C bus

48.7.11  I2C transmit data register (I2C_TXDR)

Address offset: 0x28
Reset value: 0x0000 0000
Access: no wait states

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0  \textbf{TXDATA}[7:0]: 8-bit transmit data
Data byte to be transmitted to the I2C bus
\textit{Note: These bits can be written only when TXE = 1.}
### 48.7.12 I2C register map

The table below provides the I2C register map and reset values.

| Offset | Register name   | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x0    | I2C_CR1        | STOPFAC | ADDRCF | DNF[3:0] | ERRE | STOPE | NACKF | ADDR | ADDRCF | ANFOFF | YXMAEN | TXIS | RXE | TCE | TCR | BERR | ARLO | BERRF | STOPF | ADDRF | ADDRF | TXE | RXE | NACK | ADDR | 0x0  |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x4    | I2C_CR2        | PECBYTE | AUTOEND | NBYTES[7:0] | NACK | STOP | HEAD10R | ADD10 | ADDR | ADDRF | RXDMAEN | TXDMAEN | SADD[9:0] | 0x0  |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x8    | I2C_OAR1       | OATMODE | OATMODE | OATMODE | OATMODE | OA1[9:0] | 0x0  |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x10   | I2C_TIMINGR    | TEXTEN | TEXTEN | TEXTEN | TEXTEN | TEXTEN | 0x0  |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x14   | I2C_TIMEOUTR   | TEXEN | TEXEN | TEXEN | TEXEN | TEXEN | 0x0  |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x18   | I2C_ISR        | ADDCODE[8:0] | OR | BUSY | ALERT | TIMEOUTC | TIMEOUT | TIME | TIMEOUTA | 0x0  |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x1C   | I2C_ICR        | 0x0  |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x20   | I2C_PECR       | PEC[7:0] | 0x0  |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x24   | I2C_RXDR       | RXDATA[7:0] | 0x0  |
| Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
Refer to *Section 2.3 on page 111* for the register boundary addresses.

### Table 514. I2C register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x28   | I2C_TXDR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to *Section 2.3 on page 111* for the register boundary addresses.
49  Improved inter-integrated circuit (I3C)

49.1 Introduction

The I3C interface handles communication between this device and others, such as sensors and host processor, connected on an I3C bus.

An I3C bus is a two-wire, serial single-ended, multidrop bus, intended to improve a legacy I2C bus.

The I3C SDR-only peripheral implements all the features required by the MIPI® I3C specification v1.1. It can control all I3C bus-specific sequencing, protocol, arbitration, and timing, and can act as controller (formerly known as master), or as target (formerly known as slave).

When acting as controller, the I3C peripheral improves the features of the I2C interface preserving some backward compatibility: it allows an I2C target to operate on an I3C bus in legacy I2C fast-mode (Fm) or legacy I2C fast-mode plus (Fm+), provided that the latter does not perform clock stretching.

The I3C peripheral can be used with DMA, to off-load the CPU.

49.2 I3C main features

The I3C peripheral supports:

- MIPI® I3C specification v1.1 (see details in Table 518), as:
  - I3C SDR-only primary controller
  - I3C SDR-only secondary controller
  - I3C SDR-only target
- I3C SCL bus clock frequency up to 12.5 MHz
- Registers configuration from the host application via the APB slave port
- Queued data transfers:
  - Transmit FIFO (TX-FIFO) for data bytes/words to be transmitted on the I3C bus
  - Receive FIFO (RX-FIFO) for received data bytes/words on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Queued control/status transfers, when controller:
  - Control FIFO (C-FIFO) for control words to be sent on the I3C bus
  - Optional status FIFO (S-FIFO) for status words as received on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Messages:
  - Legacy I2C read/write messages to legacy I2C targets in Fm/Fm+
  - I3C SDR read/write private messages
  - I3C SDR broadcast CCC messages (see details in Table 524)
  - I3C SDR read/write direct CCC messages (see details in Table 524)
• Frame-level management, when controller:
  – Optional C-FIFO and TX-FIFO preload
  – Multiple messages encapsulation
  – Optional arbitrable header generation on the I3C bus
  – HDR exit pattern generation on the I3C bus for error recovery
• Programmable bus timing, when controller:
  – SCL high and low period
  – SDA hold time
  – Bus free (minimum) time
  – Bus available/idle condition time
  – Clock stall time
• Target-initiated requests management:
  – Simultaneous support up to four targets, when controller
  – In-band interrupts, with programmable IBI payload (up to 4 bytes), with pending read notification support
  – Bus control request, with recovery flow support and hand-off delay
  – Hot-join mechanism
• HDR exit pattern detection, when target
• Bus error management:
  – CEx with x = 0, 1, 2, 3 when controller
  – TEx with x = 0, 1, ..., 6 when target
  – Bus control switch error and recovery
  – Target reset
• Individual programmable event-based management:
  – Per-event identification with flag reporting and clear control
  – Host application notification via flag polling, and/or via interrupt with a per-event programmable enable
  – Error type identification
• Wake-up from Stop mode(s), as controller (see Section 49.3.2):
  – On an in-band interrupt without payload
  – On a hot-join request
  – On a controller-role request
• Wakeup from Stop mode(s), as target (see Section 49.3.2):
  – On a reset pattern
  – On a missed start
• Multiclock domain management:
  – Separate APB clock and kernel clock, driven from independently programmed clock sources via the RCC, in addition to SCL clock
  – Minimum operating frequency for the kernel clock and the APB clock vs. the application-driven SCL clock (see clocks constraints in Section 49.6.2)
49.3  I3C implementation

49.3.1  I3C instantiation

There is a single I3C instance in the device.

49.3.2  I3C wake-up from low-power mode(s)

The peripheral can wake up the device from a low-power mode, as detailed in Table 515. For more details about the wake-up capabilities, refer to Section 49.13.

<table>
<thead>
<tr>
<th>Table 515. I3C wake-up</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Wake-up</strong></td>
</tr>
<tr>
<td>From Stop mode with SVOS3</td>
</tr>
</tbody>
</table>

49.3.3  I3C FIFOs

The FIFOs are implemented as defined in Table 516.

<table>
<thead>
<tr>
<th>Table 516. I3C FIFOs implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FIFO</strong></td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>C-FIFO</td>
</tr>
<tr>
<td>S-FIFO</td>
</tr>
<tr>
<td>TX-FIFO</td>
</tr>
<tr>
<td>RX-FIFO</td>
</tr>
</tbody>
</table>

49.3.4  I3C triggers

This feature is not available in this product: no hardware trigger signal is connected as an input to the I3C peripheral.

49.3.5  I3C interrupt(s)

The interrupt mapping is implemented as detailed in Table 517.

<table>
<thead>
<tr>
<th>Table 517. I3C interrupt(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal name</strong></td>
</tr>
<tr>
<td>i3c_err_it</td>
</tr>
<tr>
<td>i3c_evt_it</td>
</tr>
</tbody>
</table>
### 49.3.6 I3C MIPI® support

The I3C peripheral supports the MIPI specification v1.1, as defined in Table 518.

<table>
<thead>
<tr>
<th>Feature</th>
<th>MIPI I3C v1.1</th>
<th>I3C peripheral</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C SDR message</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Legacy (i^2C) message (Fm/Fm+)</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Mandatory when controller, and the I3C bus is mixed with (external) legacy (i^2C) target(s). Optional in MIPI v1.1 when target.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HDR DDR message</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HDR-TSL/TSP, HDR-BT</td>
<td>X</td>
<td>-</td>
<td>Optional in MIPI v1.1</td>
</tr>
<tr>
<td>Dynamic address assignment</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Static address</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>No (intended) support of the peripheral as a target on an (i^2C) bus.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grouped addressing</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Optional in MIPI v1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCCs</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Mandatory and some optional CCCs supported (refer to Table 524 when controller/target).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error detection and recovery</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>In-band interrupt (with MDB)</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Secondary controller</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Hot-join mechanism</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Target reset</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Synchronous timing control</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Optional in MIPI v1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asynchronous timing control 0</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Mandatory in MIPI v1.1 when controller. Optional in MIPI v1.1 when target.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asynchronous timing control 1, 2, 3</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Optional in MIPI v1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device to device tunneling</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Optional in MIPI v1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-lane data transfer</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Optional in MIPI v1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monitoring device early termination</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Optional in MIPI v1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
49.4 I3C block diagram

The I3C block diagram is illustrated in Figure 651.

Figure 651. I3C block diagram

1. This feature is implementation-dependent, and can be unavailable. Refer to Section 49.3.4: I3C triggers.

49.5 I3C pins and internal signals

Table 519. I3C input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_SDA</td>
<td>Bidirectional</td>
<td>I3C bus serial data line</td>
</tr>
<tr>
<td>I3C_SCL</td>
<td>Bidirectional</td>
<td>I3C bus serial clock line</td>
</tr>
</tbody>
</table>

Table 520. I3C internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i3c_pclk</td>
<td>I</td>
<td>APB clock</td>
</tr>
<tr>
<td>i3c_ker_clk</td>
<td>I</td>
<td>Kernel clock (also named as I3CCLK)</td>
</tr>
<tr>
<td>i3c_pclk_req</td>
<td>O</td>
<td>APB clock request</td>
</tr>
<tr>
<td>i3c_ker_clk_req</td>
<td>O</td>
<td>Kernel clock request</td>
</tr>
<tr>
<td>i3c_it(1)</td>
<td>O</td>
<td>Global interrupt line</td>
</tr>
</tbody>
</table>
49.6 I3C reset and clocks

49.6.1 I3C reset

On a system reset, the I3C peripheral is reset.

Alternatively, the software can reset specifically the peripheral by writing the corresponding reset control bit (I3CxRST) of the reset and clock controller (RCC). Refer to the RCC section of this document for more details.

Additionally, when acting as target, the enabled peripheral (EN = 1 in the I3C_CFGR register) can receive an in-band reset pattern on the I3C bus from the controller. The software is then notified (when RSTF = 1 in the I3C_EVR register and/or the corresponding interrupt is enabled) to perform the requested action, as registered in RSTACT[1:0] of the I3C_DEVR0 register, on the former reception of the broadcast or direct RSTACT CCC. Refer to Table 524 and Section 49.16.16 for more details.

This reset interrupt notification can be used to wake up from a low power mode, where the I3C peripheral (typically in the VCore domain) is active.

For more details about the corresponding low-power mode(s), refer to the power management in the PWR section.

49.6.2 I3C clocks and requirements

As indicated in the Figure 651, the I3C peripheral is implemented with several clock domains:

- SCL bus clock: for the I3C bus interface
  - When controller: the user must set and can adjust SCL/SDA timings by programming I3C timing register 0 (I3C_TIMINGR0), I3C timing register 1 (I3C_TIMINGR1) and I3C timing register 2 (I3C_TIMINGR2), as summarized in Controller initialization and Updating the configuration for a transfer, as controller.
  - When target: the user must set and comply with the bus available condition (tAVAL for an in-band interrupt or controller-role request), and the bus idle condition (tIDLE for a hot-join request), by programming I3C timing register 1 (I3C_TIMINGR1), as summarized in Target initialization.
- I3CCLK kernel clock: for the I3C protocol management, data and control serialization/deserialization, controller and target finite state machines, bus clock and timings management
- APB clock: for the APB interface, DMA interface, events, and interrupt generation

APB clock and kernel clocks are driven from independently programmed clock sources via the RCC (refer to Section 11: Reset and clock control (RCC)).

I3C kernel clock requirement, as controller

According to the intended value of the SCL clock on the bus, the application must guarantee that the frequency of the I3CCLK kernel clock be at least 2x the frequency of the SCL clock

*Note:* Sustaining $F_{SCL_{max}} = 12.9$ MHz means a frequency of the I3CCLK kernel clock > 25.8 MHz.

I3C kernel clock requirements, as target

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the I3CCLK kernel clock, meeting the following constraints:

1. Period of the I3CCLK kernel clock < $t_{HIGH}$ (SCL clock high period)
   - $t_{HIGH \ min} = 24$ ns. A frequency higher than 41.7 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
2. Period of the I3CCLK kernel clock < $t_{CASr}$ (clock after repeated start condition)
   - $t_{CASr \ min} = t_{CAS \ min} / 2 = 19.2$ ns. A frequency higher than 52 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
3. Two periods of the I3CCLK kernel clock < $t_{LOW\_OD}$ (SCL clock low period in open drain)
   - $t_{LOW\_OD \ min} = 200$ ns. A frequency higher than 10 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
4. Frequency of the I3CCLK kernel clock > 2.5x frequency of the SCL clock
   - $F_{SCL_{max}} = 12.9$ MHz. A frequency higher than 32.3 MHz guarantees this constraint, which can be relaxed, depending on the I3C controller

APB clock requirement

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the APB clock:

- APB clock period < 3x (SCL clock period) - I3CCLK kernel clock period

This means that $F_{APB} > [F_{SCL} \times F_{I3CCLK} / (3 \times F_{I3CCLK} - F_{SCL})]

*Note:* This equation can be simplified to a minimum value of 5 MHz for the APB frequency.
49.7 I3C peripheral state and programming

49.7.1 I3C peripheral state

The I3C peripheral plays the role of I3C bus controller, or the role of an I3C target. In any case (see Figure 652 and Figure 653) the peripheral is in one of the following states:

- **Disabled state:**
  - After an I3C reset (system reset or I3C reset from RCC), the peripheral is in disabled state.
  - When the software sets to 1 bit EN in the I3C_CFGR register, the peripheral takes into account the value of the different configuration registers, and switches to the (enabled and) idle state.

- **Idle state:**
  - After being enabled (EN = 1), the peripheral activates its I3CCLK and SCL clock domains, and is able to communicate on the I3C bus.
  - The software can partly update the I3C peripheral configuration, see Updating the configuration for a transfer, as controller and Updating the configuration of the I3C peripheral, as target for more details.
  - The peripheral switches to the (enabled and) active state, either:
    a) Once the software initiates a transfer (as controller: when the software initiates a frame transfer; as target: when the software initiates an IBI/CR/HJ request)
    b) Or once the hardware receives a request from another I3C device on the bus (as controller: after a start request from a target and a maximum TCAS time; as target: when receiving a broadcast/direct CCC or a private read/write).

- **Active state:**
  - The peripheral executes the transfer(s) on the bus
  - When the requested transfer(s) is(are) completed, the software is notified by an event from the I3C event register (I3C_EVR), and the corresponding interrupt is enabled by I3C interrupt enable register (I3C_IER). The peripheral switches to idle state, is still able to communicate on the bus, and can be (partly) reconfigured.
    a) As controller: the raised event/flag can be frame completed (FCF), IBI/controller-role/hot-join request completed (IBIF/CRF/HJF), or transfer error (ERRF)
    b) As target: the raised event/flag can be dynamic address assignment completed (DAUPDF), IBI completed (IBIENDF), controller-role gaining completed (CRUPDF), broadcast/direct CCC completed (xxUPDF/RSTF/GETF/STAF), private read/write completed (FCF), or transfer error (ERRF)

**Note:** The software can disable the peripheral (write EN = 0), partially resetting it (subparts within the SCL clock domain and the I3CCLK kernel clock domain). Event, interrupt, and clock request generation are also impacted. The previously written configuration of the APB registers is kept and not modified.

49.7.2 I3C controller state and programming sequence

*Figure 652* illustrates the overall programming sequence of the I3C peripheral acting as (primary) controller, including state transitions, main subtasks, and conditions, as explained in this section.
1. This feature is implementation-dependent and can be unavailable. Refer to Section 49.3.4: I3C triggers.
Controller initialization

When the controller is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Configure **I3C configuration register (I3C_CFGR)** with the following fields:
  - CRINIT = 1: as I3C bus controller
  - HKSDAEN: high keeper on SDA enable/disable
- Configure I3C bus timings:
  a) **I3C timing register 0 (I3C_TIMINGR0)**:
     - SCL clock high time period (tDIG_H, tDIG_H_MIXED) in legacy I²C and I3C open-drain/push-pull
     - SCL clock low time period (tDIG_L, tDIG_OD_L) in legacy I²C and I3C open-drain/push-pull
  b) **I3C timing register 1 (I3C_TIMINGR1)**:
     - SDA hold time in push-pull (tHD_PP)
     - bus free condition time (I3C tCAS, legacy I²C tBUF)
     - I3C repeated start timing (tCASr, tCBSr)
     - I3C stop timing (tCBP)
     - SCL clock low maximum stalling on the ENTDAA CCC (tSTALDDAA), or on the ACK/NACK of a legacy I²C transfer, or the parity bit of a write data transfer, or on the ACK/NACK data phase of a legacy I²C write, or on the transition bit of an I3C read transfer, or on the ACK/NACK phase of a legacy I²C write (tSTALL), to adjust SCL clock low stalling, if needed by the peripheral itself, when used as controller
     - tNEWCRLock for controller-role hand-off procedure (after GETACCCR CCC)
  c) **I3C timing register 2 (I3C_TIMINGR2)**:
     - SCL clock low stalling time, with separated enable/disable for each phase, to adjust SCL clock low stalling, if needed on the SDA hand-off with the addressed I3C target or legacy I²C target
- Configure its own dynamic address: DA [6:0] field of the **I3C own device characteristics register (I3C_DEVR0)**
- Configure the management of any device target x: **I3C device x characteristics register (I3C_DEVRx)**, for x = 1 to x ≤ 4
- Configure the execution mode of a frame transfer or a target-requested transfer: **I3C configuration register (I3C_CFGR)**, with the following fields:
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN: DMA mode enable/disable for respectively, TX-FIFO, C-FIFO, RX-FIFO, S-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - TMODE: transmit mode (enable/disable for both TX-FIFO and C-FIFO preload)
  - SMODE: S-FIFO enable/disable
  - EXITPTRN, RSTPTRN: exit, reset pattern enable/disable
  - HJACK: hot-join acknowledge enable/disable
  - NOARBH: arbitrable header disable/enable
- Configure interrupt generation or polling mode from any event: **I3C interrupt enable register (I3C_IER)**
Then, the software can enable the I3C peripheral (set EN = 1)

Note: The software can write once all the fields of the I3C_CFGR while enabling it.

**Start a controller-initiated frame transfer**

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), the software can initiate a frame transfer by any of the following configuration methods:

1. Software-triggering: on a write and set TSFSET = 1 in the I3C_CFGR register
   - This causes the hardware to raise the flag CFNFF = 1 in the I3C_EVR register, to request a first control word I3C_CR to be written.
2. No triggering: on a write of the first control word I3C_CR by software.

Then, regardless of the frame starting method, the I3C peripheral switches to active state. While the control word is not the last message of the I3C frame (while MEND = 0 in the I3C_CR register), and while there is no transfer error (while ERRF = 1 in the I3C_EVR register), the hardware keeps requesting a next control word, and continuing the frame transfer:

a) If the C-FIFO is not configured in DMA mode (CDMAEN = 0 in the I3C_CFGR register), the software writes a next control word following the flag CFNFF = 1 in the I3C_EVR register, or the corresponding interrupt if enabled (if CFNFIE = 1 in I3C_IER register)

b) If the C-FIFO is configured in DMA mode (CDMAEN = 1), a next control word is automatically pushed and written by the allocated DMA channel consequently to the asserted I3C DMA request (i3c_tc_dma).

**Start and receiving a target-initiated transfer**

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), concurrently to a possible controller-initiated transfer, a target can initiate a transfer by issuing a start request (drive SDA low), provided the controller has allowed a hot-join request, an IBI request, or a controller-role request via the I3C_DEVR0 register.

In this case, even though the controller software has no intent to start a frame transfer, the hardware switches to active state (activates the SCL clock before a maximum tCAS time defined as 1 µs, 100 µs, 2 ms or 50 ms, depending on, respectively, the bus activity state 0, 1, 2 or 3) to receive the hot-join/in-band interrupt/controller-role request from the target.

For more information about the execution of target-initiated I3C bus transfer and its related programming as a controller, refer to the relevant figures in Section 49.9:

- Figure 664: IBI transfer, as controller/target
- Figure 665: Hot-join request transfer, as controller/target
- Figure 666: Controller-role request transfer, as controller/target

**Executing a (controller-initiated) frame transfer**

The controller executes on the bus the frame transfer until the completion of the last message (FCF = 1 in the I3C_EVR register), or a transfer error (ERRF = 1 in the I3C_EVR register), and the corresponding interrupt, if enabled. This is based on I3C_CR and I3C_TD(W)R registers, written explicitly by software or pushed by the allocated DMA channel, and based on the I3C_RD(W)R, read explicitly by the software or by the allocated DMA channel. Then the I3C controller switches back to idle state.
For more information about the execution of controller-initiated I3C bus transfer and its related programming as a controller, refer to figures in Section 49.9:

- Figure 654: I3C CCC messages, as controller
- Figure 655: I3C broadcast ENTDAA CCC, as controller
- Figure 656: I3C broadcast, direct read and direct write RSTACT CCC, as controller
- Figure 661: I3C private read/write messages, as controller
- Figure 663: Legacy I2C read/write messages, as controller

Figure 652 does not include the management of the FIFOs (TX-FIFO, RX-FIFO, C-FIFO, and S-FIFO). This is detailed in Section 49.10.

For each completed message without transfer error, the hardware reports the exchanged transfer on the I3C bus by updating I3C status register (I3C_SR), which can be read or not by the software when the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register).

- In the case of a direct CCC read or a private read transfer, in addition to the completion of the last message (FCF = 1 in the I3C_EVR register) or a transfer error (ERRF = 1 in the I3C_EVR register) and the corresponding interrupt if enabled, and provided that the S-FIFO is disabled for the status register I3C_SR (SMODE = 0 in the I3C_CFGR register), the software is notified if the read transfer is ended prematurely by the target by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt, if enabled. The software can then read I3C_SR, to get more information about the executed transfer.

Alternatively, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the status register I3C_SR must be read for each executed message, either directly by the software (notified by SFNEF = 1 in the I3C_EVR register and the corresponding interrupt, if enabled), or via the DMA (if SDMAEN = 1 in the I3C_CFGR register), no matter if a read is prematurely ended by the target or not. Frame completion (FCF = 1 in the I3C_EVR register) occurs only after reading the status of the last message (S-FIFO is empty). For more information, refer to Section 49.10.4.

**Updating the configuration for a transfer, as controller**

Back in idle state, the software can update the configuration of the I3C peripheral before the next transfer:

- Modify SCL clock stalling via **I3C timing register 2 (I3C_TIMINGR2)**
- Modify the interrupt/polling mode policy via **I3C interrupt enable register (I3C_IER)**
- Modify the following fields of the **I3C configuration register (I3C_CFGR):**
  - TXTHRES, RXTHRES
  - TMODE, SMODE
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN
  - EXITPTRN, RSTPTRN
  - NOARBH
- Modify/prepare the control words, status words, read/write data of the next frame transfer to be executed, by software and/or DMA
  - \textit{I3C message control register (I3C\_CR), I3C message control register [alternate] (I3C\_CR)}
  - \textit{I3C status register (I3C\_SR)}
  - \textit{I3C transmit data byte register (I3C\_TDR), I3C transmit data word register (I3C\_TDWR)}
  - \textit{I3C receive data byte register (I3C\_RDR), I3C receive data word register (I3C\_RDWR)}

- Typically after having issued and completed a broadcast/direct DISEC/ENEC CCC:
  - Modify the hot-join acknowledge policy via bit HJACK in the I3C\_CFGR register
  - Modify IBI/CR acknowledge policy to any target x, via \textit{I3C device x characteristics register (I3C\_DEVRx)}

The registers usage vs. the I3C peripheral role as controller is summarized in Section 49.8.1.

The static/dynamic registers fields usage when acting as controller is summarized in Table 522.

### 49.7.3 I3C target state and programming sequence

\textit{Figure 653} illustrates the overall programming sequence of the peripheral acting as target, including state transitions, main subtasks, and conditions as explained in this section.
Figure 653. I3C target state and programming sequence diagram
Target initialization

When the target is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Set I3C peripheral as target: write and clear CRINIT = 0 in the I3C_CFGR register
- Set I3C bus timings via I3C timing register 1 (I3C_TIMINGR1): write AVAL[7:0] for:
  - Bus available condition time (tAVAL) for IBI or controller-role request
  - Bus idle condition time (tIDLE) for hot-join request
  - tNEWCRLock for controller-role hand-off procedure (after GETACCCR CCC)
- Configure target-initiated requests: write I3C own device characteristics register (I3C_DEVR0) with
  - IBIEN: in-band interrupt (also known as IBI) request enable/disable
  - CREN: controller-role request enable/disable
  - HJEN: hot-join request enable/disable
- Initialize target characteristics and capabilities: write
  - I3C bus characteristics register (I3C_BCR)
  - I3C device characteristics register (I3C_DCR)
  - I3C maximum read length register (I3C_MAXRLR)
  - I3C maximum write length register (I3C_MAXWLR)
  - I3C get capability register (I3C_GETCAPR)
  - I3C controller-role capability register (I3C_CRCAPR)
  - I3C get max data speed register (I3C_GETMXDSR)
  - I3C extended provisioned ID register (I3C_EPIDR)
- Configure the execution mode of a transfer: I3C configuration register (I3C_CFGR), with the following fields:
  - TXDMAEN, RXDMAEN: DMA mode enable/disable for, respectively, TX-FIFO and RX-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - Disable S-FIFO: SMODE = 0 (default/reset value)
- Configure interrupt generation or polling mode from any event: I3C interrupt enable register (I3C_IER)

Then, the software can enable the I3C peripheral (write and set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR register while enabling it.

Receiving a (broadcast CCC, direct read/write CCC or private read/write) message from the controller

When the target is in idle state (EN = 1 in the I3C_CFGR register), the target is ready to receive a communication message on the I3C bus from the controller, and is ready to switch to the active state.

Typically, first, the active target is receiving a broadcast ENTDAA CCC, possibly after optional received broadcast ENEC/DISEC CCC(s), and is then assigned a dynamic address. The event DAUPF in the I3C_EVR register is raised to 1, the related interrupt is generated if enabled, and the target goes back to idle state.
After that, the idle target is ready to receive any other broadcast CCC message, or direct read/write CCC, or private read/write message from the controller.

For more information about the execution of controller-initiated I3C bus transfers and its related programming as a target, including the updated I3C registers and fields, refer to figures in Section 49.9:

- Figure 657: I3C CCC messages, as target
- Figure 658: I3C broadcast ENTDAA CCC, as target
- Figure 659: I3C broadcast DEFTGTS CCC, as target
- Figure 660: I3C broadcast DEFR GPA CCC, as target
- Figure 662: I3C private read/write messages, as target

Figure 653 does not include the FIFOs management (TX-FIFO, RX-FIFO). This is detailed in Section 49.10.

Read the message status register

For each received and completed message without transfer error, the hardware reports the exchanged transfer on the I3C bus by updating the I3C status register (I3C_SR), which can be read by the software after being notified by the corresponding flag in the I3C event register (I3C_EVR), or by the corresponding interrupt if enabled in the I3C interrupt enable register (I3C_IER).

I3C status register (I3C_SR) must be read by the software after the following messages:

- A private read: to get the number of exchanged data bytes, as the controller can have ended the transfer earlier than expected by the target (if XDCNT[15:0] in the I3C_SR register is lower than TGTTDCNT[15:0] in the I3C_TGTTDR register). If so, software must flush the TX-FIFO (write TXFLUSH = 1 in the I3C_CFGR register).
- A DEFTGTS CCC or a DEFR GPA CCC: to get the number of received data bytes in the RX-FIFO

Start a (target-initiated) transfer

When the target goes first from disabled to idle state (software writes EN = 1 in the I3C_CFGR register), concurrently to be able to receive a broadcast CCC from the controller, the software can initiate a hot-join request (the software writes MTYPE[3:0] = 1000 in the I3C_CR register) to be eligible to participate to a next ENTDAA CCC, provided it is allowed to do so (HJEN = 1 in the I3C_DEVR0 register).

Once a dynamic address is assigned (DAUPF = 1 in the I3C_EVR register), more generally and possibly concurrently to a frame transfer emitted by the controller, the software can initiate an IBI (in-band interrupt request) to the controller, or a controller-role request by writing the related control word into the I3C_CR register.

For more information about the execution of target-initiated I3C bus transfer, and its related programming as a target, refer to figures in Section 49.9:

- Figure 664: IBI transfer, as controller/target
- Figure 665: Hot-join request transfer, as controller/target
- Figure 666: Controller-role request transfer, as controller/target
Updating the configuration of the I3C peripheral, as target

Back in idle state, the software can update the configuration of the I3C target before a next transfer:

- Modify the interrupt/polling mode policy via I3C interrupt enable register (I3C_IER)
- Modify following fields of the I3C configuration register (I3C_CFGFR):
  - TXTHRES, RXTHRES
  - TXDMAEN, RXDMAEN
- Modify/prepare the I3C IBI payload data register (I3C_IBIDR), if any payload (if BCR2 = 1 in the I3C_BCR register), before initiating an IBI transfer (write I3C message control register [alternate] (I3C_CR) with MTYPE[3:0] = 1010)
- Modify/prepare I3C target transmit configuration register (I3C_TGTTDR), to disable or enable the TX-FIFO to be preloaded with a defined number of data bytes to be transmitted, before receiving a private read or a direct CCC read (out of the GETSTATUS CCC) from the controller

The registers usage vs. the I3C peripheral role as target is summarized in Section 49.8.1.

The static/dynamic registers fields usage, when acting as target, is summarized in Table 523.

49.8 I3C registers and programming

49.8.1 I3C register set, as controller/target

Table 521 lists the registers and their usage versus the I3C peripheral role.

<table>
<thead>
<tr>
<th>Register</th>
<th>Used as controller</th>
<th>Used as target</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_CR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_CFGFR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_RDR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_RDWR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_TDR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_TDWR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_IBIDR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_TGTTDR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_SR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_SER</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_RMR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_EVR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_IER</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_CEVR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_DEVR0</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
49.8.2 I3C registers and fields use vs. peripheral state, as controller

When the I3C peripheral acts as controller, Table 522 lists the registers and their usage versus the controller state (disabled, idle, and active).

<table>
<thead>
<tr>
<th>Register</th>
<th>Used as controller</th>
<th>Writable only in disabled state</th>
<th>Typically written/read in idle state</th>
<th>Typically written/read in idle or active state</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_CR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 521. I3C register usage (continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>Used as controller</th>
<th>Used as target</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_DEVRx x = 1..4</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>I3C_MAXRLR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_MAXWLR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_TIMINGR0</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TIMINGR1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_TIMINGR2</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>I3C_BCR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_DCR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_GETCAPR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_CRCAPR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_GETMXDSR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_EPIDR</td>
<td>-</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 522. I3C registers/fields usage versus controller state
### Table 522. I3C registers/fields usage versus controller state (continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>Used as controller</th>
<th>Writable only in disabled state</th>
<th>Typically written/read in idle state</th>
<th>Typically written/read in idle or active state</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_CFRG</td>
<td>X</td>
<td>CRINIT HKSDAEN EN(1)</td>
<td></td>
<td>Write: any used field except (CRINIT, HKSDAEN), namely (CDMAEN SDMAEN TXDMAEN RXDMAEN TMODE SMODE TXTHRES RXTHRES HJACK EXITPTRN RSTPTRN NOARBH)(^{(2)}) CFLUSH SFLUSH TXFLUSH RXFLUSH TSFSET</td>
</tr>
<tr>
<td>I3C_RDR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_RDWR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TDR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TDWR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_IBIDR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TGTTDR</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_SR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_SER</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_RMR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Read RADD[6:0] IBIRDCNT[2:0]</td>
</tr>
</tbody>
</table>
### Table 522. I3C registers/fields usage versus controller state (continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>Used as controller</th>
<th>Writable only in disabled state</th>
<th>Typically written/read in idle state</th>
<th>Typically written/read in idle or active state</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_EVR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Read (controller-role fields): HJF, CRF, IBIF, FCF, ERF, RXTGTENDF, RXFNEF, TXFNF, SFNEF, CFNF, RXLASTF, TXLASTF, TXFEF, CFEF</td>
</tr>
<tr>
<td>I3C_IER</td>
<td>X</td>
<td>Write xIE with x = HJ, CR, IBI, FC, ERR, RXTGTEND, RXFNE, TXFNF, SFNE, CFNF(2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I3C_CEVR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Write (controller-role fields, refer to I3C_EVR)</td>
</tr>
<tr>
<td>I3C_DEVR0</td>
<td>X</td>
<td>Write DA<a href="2">6:0</a></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_DEVRx x = 1..4</td>
<td>X</td>
<td>Write SUSP, IBIDEN, IBIACK, CRACK, DA<a href="2">6:0</a></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>I3C_MAXRLR</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_MAXWLR</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TIMINGR0</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TIMINGR1</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TIMINGR2</td>
<td>X</td>
<td>Write(2)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_BCR</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_DCR</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_GETCAPR</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_CRCAPR</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_GETMXDSR</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_EPIDR</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1. Bit EN in the I3C_CFRG register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.

2. These fields are typically written and initialized in disabled state during bus configuration. They are not write-protected when EN = 0, and can be also written and updated in other state(s).
### 49.8.3 I3C registers and fields usage vs. peripheral state, as target

When the peripheral acts as target, Table 523 lists the registers and their usage versus the I3C target state (disabled, idle, and active).

#### Table 523. I3C registers/fields usage versus target state

<table>
<thead>
<tr>
<th>Register</th>
<th>Used as target</th>
<th>Writable only in disabled state</th>
<th>Typically written/read in idle state</th>
<th>Typically written/read in idle or active state</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_CR</td>
<td>X</td>
<td>-</td>
<td>MTYPE[3:0]</td>
<td>DCNT[2:0]</td>
</tr>
<tr>
<td>I3C_CFGR</td>
<td>X</td>
<td>CRINIT</td>
<td>Write: any used field except CRINIT, namely: {TXDMAEN RXDMAEN TXTHRES RXTHRES})</td>
<td>-</td>
</tr>
<tr>
<td>I3C_RDR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Read</td>
</tr>
<tr>
<td>I3C_RDWR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Read</td>
</tr>
<tr>
<td>I3C_TDR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Write</td>
</tr>
<tr>
<td>I3C_TDWR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Write</td>
</tr>
<tr>
<td>I3C_IBIDR</td>
<td>X</td>
<td>-</td>
<td>Write</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TGTTDR</td>
<td>X</td>
<td>-</td>
<td>Read DIR, XDCNT[15:0]</td>
<td>Read RCODE[7:0]</td>
</tr>
<tr>
<td>I3C_SR</td>
<td>X</td>
<td>-</td>
<td>Read DOVR, STALL, PERR, CODERR[3:0]</td>
<td>-</td>
</tr>
<tr>
<td>I3C_SER</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### Table 523. I3C registers/fields usage versus target state (continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>Used as target</th>
<th>Writable only in disabled state</th>
<th>Typically written/read in idle state</th>
<th>Typically written/read in idle or active state</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_EVR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Read (target-role fields): GRPF, DEFF, INTUPDF, ASUPDF, RSTF, MRLUPDF, MWLUPDF, DAUPDF, STAF, GETF, WKP, CRUPDF, IBIENDF, ERRF, FCF, RXFNEF, TXFNF, TXLASTF, TXFEF.</td>
</tr>
<tr>
<td>I3C_IEER</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Write xIE with x = GRP, DEF, INTUPD, ASUPD, RST, MRLUPD, MWLUPD, DAUPD, STA, GET, WKP, CRUPD, IBIEND, ERR, FC, RXFNE, TXFNF.</td>
</tr>
<tr>
<td>I3C_CEVR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Write (target-role fields, refer to I3C_EVR)</td>
</tr>
<tr>
<td>I3C_DEVRO</td>
<td>X</td>
<td>HJEN, CREN, IBIEN</td>
<td>Read RSTVAL, RSTACT[1:0], and AS[1:0]</td>
<td>-</td>
</tr>
<tr>
<td>I3C_DEVRX</td>
<td>x = 1..4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_MAXRLR</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_MAXWLR</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TIMINGR0</td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TIMINGR1</td>
<td>X</td>
<td>AVAL[7:0]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TIMINGR2</td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_BCR</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_DCR</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_GETCAPR</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_CRCAPR</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
49.9 I3C bus transfers and programming

49.9.1 I3C command set (CCCs), as controller/target

The list of the supported I3C command set (for example, list of CCCs, common command codes) and the overview of how they are handled by the peripheral acting as controller or target, is specified in Table 524.
Table 524. List of supported I3C CCCs, as controller/target

<table>
<thead>
<tr>
<th>CCC name</th>
<th>CCC value</th>
<th>Read/write</th>
<th>With/without defining byte</th>
<th>With/without optional data byte(s)</th>
<th>Use as controller</th>
<th>Use as target, raised I3C_EV0 event</th>
<th>When target: specific action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENEC</td>
<td>0x00</td>
<td></td>
<td>With one data byte (enable target events byte)</td>
<td>X, INTUPDF</td>
<td></td>
<td>Update and enable I3C_EV0: HJEN, CREN, IBIEN if any</td>
<td></td>
</tr>
<tr>
<td>DISEC</td>
<td>0x01</td>
<td></td>
<td>With one data byte (disable target events byte)</td>
<td>X, INTUPDF</td>
<td></td>
<td>Update and disable I3C_EV0: HJEN, CREN, IBIEN if any</td>
<td></td>
</tr>
<tr>
<td>ENTASx x = 0...3</td>
<td>0x02 0x05</td>
<td></td>
<td>No data byte</td>
<td>X, ASUPDF</td>
<td></td>
<td>Update I3C_EV0.AS[1:0]</td>
<td></td>
</tr>
<tr>
<td>RSTDAA</td>
<td>0x06</td>
<td>Write</td>
<td>No defining/sub-command byte</td>
<td>X, DAUPDF</td>
<td></td>
<td>Clear I3C_EV0.DAVAL = 0</td>
<td></td>
</tr>
<tr>
<td>ENTDAA</td>
<td>0x07</td>
<td></td>
<td>Write</td>
<td></td>
<td></td>
<td>Update I3C_EV0: DA[6:0] and set DAVAL = 1</td>
<td></td>
</tr>
<tr>
<td>DEFTGTS</td>
<td>0x08</td>
<td>Write</td>
<td>With [1+ 4x (1+ number_of_targets)] x data bytes</td>
<td>X, DEFF</td>
<td></td>
<td>Update I3C_RDR/ I3C_RDWR. Refer to Figure 659.</td>
<td></td>
</tr>
<tr>
<td>SETMVL</td>
<td>0x09</td>
<td></td>
<td>With two data byte</td>
<td>X, MWLUPDF</td>
<td></td>
<td>Update I3C_MAXWLR</td>
<td></td>
</tr>
<tr>
<td>SETMRL</td>
<td>0x0A</td>
<td></td>
<td>With 2 or 3 data bytes</td>
<td>X, MRLUPDF</td>
<td></td>
<td>Update I3C_MAXRLR</td>
<td></td>
</tr>
<tr>
<td>ENTTM</td>
<td>0x0B</td>
<td></td>
<td>With one data byte</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETXTIME</td>
<td>0x28</td>
<td></td>
<td>With sub-command byte</td>
<td>Without or with one or more data bytes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETAASA</td>
<td>0x29</td>
<td></td>
<td>No defining/sub-command byte</td>
<td>No data byte</td>
<td>X</td>
<td>Update I3C_EV0: RSTF after detected reset pattern</td>
<td></td>
</tr>
<tr>
<td>RSTACT</td>
<td>0x2A</td>
<td></td>
<td>With defining byte (0x00, 0x01 or 0x02)</td>
<td>X, RSTF after detected reset pattern</td>
<td></td>
<td>Update I3C_EV0: RSTACT[1:0] and set RSTVAL = 1</td>
<td></td>
</tr>
<tr>
<td>DEFGRPA</td>
<td>0x2B</td>
<td></td>
<td>No defining/sub-command byte</td>
<td>With several data bytes</td>
<td>X, GRPF</td>
<td>Update I3C_RDR/ RDWR. Refer to Figure 660.</td>
<td></td>
</tr>
<tr>
<td>RSTGPRPA</td>
<td>0x2C</td>
<td></td>
<td>No defining/sub-command byte</td>
<td>No data byte</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Note: X represents a required byte.*
## Table 524. List of supported I3C CCCs, as controller/target (continued)

<table>
<thead>
<tr>
<th>CCC name</th>
<th>CCC value</th>
<th>Read /write</th>
<th>With/without defining byte</th>
<th>With/without optional data byte(s)</th>
<th>Use as target, raised I3C_EVR event</th>
<th>When target: specific action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Action if ACK (if I3C target Address = I3C_DEVR0.DA[6:0] and I3C_DEVR0.DAVAL = 1) (else NACK)</td>
</tr>
<tr>
<td>Direct CCCs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENEC</td>
<td>0x80</td>
<td>Read</td>
<td>No defining/sub-command byte</td>
<td>No data byte</td>
<td>X, INTUPDF</td>
<td>Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any</td>
</tr>
<tr>
<td>DISEC</td>
<td>0x81</td>
<td></td>
<td>No defining/sub-command byte</td>
<td>No data byte</td>
<td>X, INTUPDF</td>
<td>Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any</td>
</tr>
<tr>
<td>ENTASx</td>
<td>0x82</td>
<td>Write</td>
<td>No defining/sub-command byte</td>
<td>No data byte</td>
<td>X, ASUPDF</td>
<td>Update I3C_DEVR0.AS[1:0]</td>
</tr>
<tr>
<td>SETDASA</td>
<td>0x87</td>
<td></td>
<td>No data byte</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETNEWDA</td>
<td>0x88</td>
<td></td>
<td>With one data byte</td>
<td>X, DAUPDF</td>
<td>Update I3C_DEVR0: DA[6:0] (and set DAVAL = 1)</td>
<td></td>
</tr>
<tr>
<td>SETMWL</td>
<td>0x89</td>
<td></td>
<td>With two data bytes</td>
<td>X, MWLUPDF</td>
<td>Update I3C_MAXWLR</td>
<td></td>
</tr>
<tr>
<td>SETMRL</td>
<td>0x8A</td>
<td></td>
<td>With two or three data bytes</td>
<td>X, MRLUPDF</td>
<td>Update I3C_MAXRLR</td>
<td></td>
</tr>
<tr>
<td>GETMWL</td>
<td>0x8B</td>
<td>Read</td>
<td>No defining/sub-command byte</td>
<td>With two data bytes</td>
<td>X, GETF</td>
<td>Return data bytes from I3C_MAXWLR[15:0]. Refer to Section 49.16.19.</td>
</tr>
<tr>
<td>GETMRL</td>
<td>0x8C</td>
<td></td>
<td>With two or three data bytes</td>
<td>With two or three data bytes</td>
<td>X, GETF</td>
<td>Return data bytes from I3C_MAXRLR[15:0] and if I3C_BCR.BCR2 = 1 return third byte from I3C_MAXRLR.IBIP[2:0]. Refer to Section 49.16.18.</td>
</tr>
<tr>
<td>CCC name</td>
<td>CCC value</td>
<td>Read/write</td>
<td>With/without defining byte</td>
<td>With/without optional data byte(s)</td>
<td>Use as controller</td>
<td>Use as target, raised I3C_EV event</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
<td>------------</td>
<td>----------------------------</td>
<td>-------------------------------------</td>
<td>-------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>GETPID</td>
<td>0x8D</td>
<td>No defining/sub-command byte</td>
<td>With six data bytes</td>
<td>X, GETF</td>
<td>Return data bytes from I3C_EPIDR. Refer to Section 49.16.28.</td>
<td></td>
</tr>
<tr>
<td>GETBCR</td>
<td>0x8E</td>
<td>No defining/sub-command byte</td>
<td>With one data byte</td>
<td>X, GETF</td>
<td>Return data byte from I3C_BCR[7:0]. Refer to Section 49.16.23.</td>
<td></td>
</tr>
<tr>
<td>GETDCR</td>
<td>0x8F</td>
<td>No defining/sub-command byte</td>
<td>With two data bytes</td>
<td>X, GETF</td>
<td>Return I3C_DCR[7:0]. Refer to Section 49.16.24.</td>
<td></td>
</tr>
<tr>
<td>GETSTATUS</td>
<td>0x90</td>
<td>With or without defining byte (TGTSTAT, PRECR)</td>
<td>With two data bytes (format 1 or 2 with PRECR)</td>
<td>X, STAF if format 1, X, GETF if format 2</td>
<td>Return 2 data bytes, as detailed in Section 49.9.9.</td>
<td></td>
</tr>
<tr>
<td>GETACCCR</td>
<td>0x91</td>
<td>No defining/sub-command byte</td>
<td>With one data byte</td>
<td>X, CRUPDF</td>
<td>Return data byte from I3C_DEVR0.DA[6:0] with parity bit</td>
<td></td>
</tr>
<tr>
<td>GETMXDS</td>
<td>0x94</td>
<td>With or without defining byte (WRRDTURN, CRHDLY)</td>
<td>With two data bytes (format 1 or 5 data bytes (format 2 or format 3 with WRRDTURN) or 1 data byte (format 3 with CRHDLY)</td>
<td>X, GETF</td>
<td>Return data byte(s) from I3C_GETMXDSR. Refer to Section 49.16.27.</td>
<td></td>
</tr>
<tr>
<td>GETCAPS</td>
<td>0x95</td>
<td>With or without defining byte (TGTSTAT, CRCAPS)</td>
<td>With 3 data bytes (format 1 or format 2 with TGTSTAT) or two data bytes (format 2 with CRCAPS)</td>
<td>X, GETF</td>
<td>Return 3 GETCAPx data bytes from I3C_GETCAPR (refer to Section 49.16.25) or Return 2 CRCAPx data bytes from I3C_CRCAPR (refer to Section 49.16.26)</td>
<td></td>
</tr>
<tr>
<td>D2DXFER</td>
<td>0x97</td>
<td>Write</td>
<td>With defining byte</td>
<td>With defining byte</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>SETXTIME</td>
<td>0x98</td>
<td>With sub-command byte</td>
<td>With sub-command byte</td>
<td>X</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>GETXTIME</td>
<td>0x99</td>
<td>Read</td>
<td>No defining/sub-command byte</td>
<td>No defining/sub-command byte</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 524. List of supported I3C CCCs, as controller/target (continued)

<table>
<thead>
<tr>
<th>CCC name</th>
<th>CCC value</th>
<th>Read/write</th>
<th>With/without defining byte</th>
<th>With/without optional data byte(s)</th>
<th>Use as controller</th>
<th>Use as target, raised I3C_EVR event</th>
<th>When target: specific action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSTACT</td>
<td>0x9A</td>
<td>Read/Write</td>
<td>With defining byte (0x00, 0x01, or 0x02)</td>
<td>With defining byte (0x00, 0x01, or 0x02)</td>
<td>X</td>
<td>X, RSTF if detected reset pattern</td>
<td>Read: return data byte from RSTACT[1:0] in the I3C_DEVR0 register. Write: update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1</td>
</tr>
<tr>
<td>SETGRPA</td>
<td>0x9B</td>
<td>Write</td>
<td>No defining/sub-command byte</td>
<td>No defining/sub-command byte</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RSTGRPA</td>
<td>0x9C</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

49.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller

*Figure 654* illustrates I3C broadcast CCC write transfer (except ENTDAA, RSTACT), and direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.
Figure 654. I3C CCC messages, as controller

---

**LEGEND**
- Controller (or target) drives SDA low/high in open-drain (arbitrable header)
- Controller drives SDA in open-drain
- Controller drives SDA in push-pull
- Target drives SDA in push-pull
- Acknowledge from the addressed target(s) which drives SDA low in open-drain
- Acknowledge from the addressed target(s) which drives SDA low in open-drain without hand-off (or continuous driving SDA)
- Sr is followed by 7'h7E broadcast address at the end of a direct CCC message
- Sr*: Transition bit (parity bit for write data) from controller (drives SDA high in push-pull)
- Sr*: Transition bit (end of data for read data) from controller and/or from target (drives SDA low in push-pull)

---

I3C broadcast CCC write (excepted ENTDAA, RSTACT)

<table>
<thead>
<tr>
<th>Sr</th>
<th>I3C Reserved Read (7'h7E) (Broadcast)</th>
<th>ACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sr</td>
<td>I3C Reserved Read (7'h7E) (Broadcast)</td>
<td>ACK</td>
</tr>
</tbody>
</table>

---

I3C direct CCC write, 1st part (excepted RSTACT)

<table>
<thead>
<tr>
<th>Sr</th>
<th>I3C Reserved Read (7'h7E) (Broadcast)</th>
<th>ACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sr</td>
<td>I3C Reserved Read (7'h7E) (Broadcast)</td>
<td>ACK</td>
</tr>
</tbody>
</table>

---

I3C direct CCC write, n x 2nd part (excepted RSTACT)

<table>
<thead>
<tr>
<th>Sr</th>
<th>I3C Target Address (RnW=1)</th>
<th>ACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sr(*) or P</td>
<td>I3C CR.MEND=0</td>
<td></td>
</tr>
<tr>
<td>Sr(*) or P</td>
<td>I3C CR.MTYPE[3:0]=0011</td>
<td></td>
</tr>
<tr>
<td>Sr(*) or P</td>
<td>I3C CR.CCC[7:0] = 0</td>
<td></td>
</tr>
<tr>
<td>Sr(*) or P</td>
<td>I3C CR.DCNT[15:0] = N</td>
<td></td>
</tr>
</tbody>
</table>

---

I3C direct CCC read, 1st part (excepted RSTACT)

<table>
<thead>
<tr>
<th>Sr</th>
<th>I3C Reserved Read (7'h7E) (Broadcast)</th>
<th>ACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sr</td>
<td>I3C Reserved Read (7'h7E) (Broadcast)</td>
<td>ACK</td>
</tr>
</tbody>
</table>

---

I3C direct CCC read, n x 2nd part (excepted RSTACT)

| Sr(*) or P | I3C CR.MEND=0 |
| Sr(*) or P | I3C CR.MTYPE[3:0]=0011 |
| Sr(*) or P | I3C CR.CCC[7:0] = 0 |
| Sr(*) or P | I3C CR.DCNT[15:0] = N |

---

Optional Write

Data-N

I3C CR.MEND

I3C CR.MTYPE[3:0]=0110

I3C CR.CCC[7:0] = 0

I3C CR.DCNT[15:0] = N

If CCC defining byte/sub-command byte/data byte(s)

I3C_TDR/TDWR

Previous transfer

ACK

Sr or P

Optional Write

Data-1

I3C CR.MEND

I3C CR.MTYPE[3:0]=0110

I3C CR.CCC[7:0] = 0

I3C CR.DCNT[15:0] = N

If CCC defining byte/sub-command byte/data byte(s)
49.9.3 I3C broadcast ENTDAA CCC transfer, as controller

Figure 655 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as controller.

49.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller

Figure 656 illustrates I3C broadcast (write), direct write and read RSTACT CCC, as communicated on the I3C bus, and as programmed when acting as controller.
Figure 656. I3C broadcast, direct read and direct write RSTACT CCC, as controller
49.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGAPA), as target

*Figure 657* illustrates I3C broadcast CCC write transfer (except ENTDAA, DEFTGTS, DEFGAPA), direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.
Figure 657. I3C CCC messages, as target

**I3C broadcast CCC write**
- When IP acts as target
- Previous transfer
  - Sr (I3C reserved byte)
  - I3C reserved byte (7'h7E)
- ACK
  - I3C_RMR.RCODE[7:0]
- Optional write Data-N
- If CCC defining byte/sub-command byte/data byte(s)
- Previous transfer
  - Sr (I3C reserved byte)
  - ACK
  - Sr (I3C reserved byte)
- Optional write Data-N
- Write Data-N
- Sr(*) or P
  - Controller drives SDA in push-pull
  - Controller drives Data-N (Sr follows 7'h7E+W broadcast address)

**I3C broadcast CCC write**
- When IP acts as target
- Previous transfer
  - Sr (I3C reserved byte)
  - I3C reserved byte (7'h7E)
- ACK
  - I3C_RMR.RCODE[7:0]
- Optional write Data-N
- If CCC defining byte/sub-command byte/data byte(s)
- Previous transfer
  - Sr (I3C reserved byte)
  - ACK
  - Sr (I3C reserved byte)
- Optional write Data-N
- Write Data-N
- Sr(*) or P
  - Controller drives SDA in push-pull
  - Controller drives Data-N (Sr follows 7'h7E+W broadcast address)

**I3C broadcast CCC read**
- When IP acts as target
- Previous transfer
  - Sr (I3C reserved byte)
  - I3C reserved byte (7'h7E)
- ACK
  - I3C_RMR.RCODE[7:0]
- Optional read Data-N
- If CCC defining byte/sub-command byte/data byte(s)
- Previous transfer
  - Sr (I3C reserved byte)
  - ACK
  - Sr (I3C reserved byte)
- Optional read Data-N
- Read Data-N
- Sr(*) or P
  - Controller drives SDA in push-pull
  - Controller drives Data-N (Sr follows 7'h7E+W broadcast address)

**I3C direct CCC write**
- When IP acts as target
- Previous transfer
  - Sr (I3C reserved byte)
  - I3C reserved byte (7'h7E)
- ACK
  - I3C_RMR.RCODE[7:0]
- Optional write Data-N
- If CCC defining byte/sub-command byte/data byte(s)
- Previous transfer
  - Sr (I3C reserved byte)
  - ACK
  - Sr (I3C reserved byte)
- Optional write Data-N
- Write Data-N
- Sr(*) or P
  - Controller drives SDA in push-pull
  - Controller drives Data-N (Sr follows 7'h7E+W broadcast address)

**I3C direct CCC read**
- When IP acts as target
- Previous transfer
  - Sr (I3C reserved byte)
  - I3C reserved byte (7'h7E)
- ACK
  - I3C_RMR.RCODE[7:0]
- Optional read Data-N
- If CCC defining byte/sub-command byte/data byte(s)
- Previous transfer
  - Sr (I3C reserved byte)
  - ACK
  - Sr (I3C reserved byte)
- Optional read Data-N
- Read Data-N
- Sr(*) or P
  - Controller drives SDA in push-pull
  - Controller drives Data-N (Sr follows 7'h7E+W broadcast address)

**LEGEND**
- Controller drives SDA in push-pull
- Controller drives SDA in open-drain
- Target drives SDA in push-pull
- Acknowledge from the addressed target(s) (Sr is followed by 7'h7E+W broadcast address)
- Acknowledge from the addressed target(s) (Sr is followed by 7'h7E+W broadcast address without final ACK)
- Transition bit (parity bit for write data) from controller (drives SDA in push-pull)
- Transition bit (parity bit for write data) without final ACK (Sr is followed by 7'h7E+W broadcast address)
- Transition bit (end of data for read data) from controller and/or from target (Sr is followed by 7'h7E+W broadcast address without final ACK)
49.9.6 I3C broadcast ENTDAA CCC transfer, as target

*Figure 658* illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as target.

*Figure 658. I3C broadcast ENTDAA CCC, as target*
49.9.7 I3C broadcast DEFTGTS CCC transfer, as target

*Figure 659* illustrates I3C broadcast DEFTGTS CCC, as communicated on the I3C bus, and as programmed when acting as target.

*Figure 659. I3C broadcast DEFTGTS CCC, as target*
49.9.8  I3C broadcast DEFGPRA CCC transfer, as target

Figure 660 illustrates I3C broadcast DEFGPRA CCC, as communicated on the I3C bus, and as programmed when acting as target.

Figure 660. I3C broadcast DEFGPRA CCC, as target

49.9.9  I3C direct GETSTATUS CCC response, as target

When the I3C acts as target, the hardware returns two data bytes on reception of GETSTATUS CCC, with format 1 (without defining byte or with defining byte TGTSTAT = 0x00), or format 2 (with defining byte PRECR = 0x91).

The returned 2-byte STATUS[15:0] with format 1 on the I3C bus is then as follows:

- STATUS[15:14] = 00 (unused)
- STATUS[13] = 1 if a missed start was detected since the former GETSTATUS CCC, else 0
- STATUS[12] = 1 if an overrun/underrun error was detected since the former GETSTATUS CCC, else 0
- STATUS[11] = 1 if an SCL stable for more than 125 µs was detected during an SDR read since the former GETSTATUS CCC, else 0
- STATUS[10:8] = 000 to 110: encoded value x = 0 to 6, corresponding to a target error TEx if a protocol error was detected since the former GETSTATUS CCC (if STATUS[5] = 1), else 000
- STATUS[7:6] = 00 (ready to prepare for hand-off procedure)
- STATUS[5] = 1 if a protocol error was detected since the former GETSTATUS CCC, else 0
- STATUS[4] = 0 (reserved)
- STATUS[3:1] = 000 (unused)
• STATUS[0] = 1 if there is a pending interrupt (if an IBI is configured in the I3C_CR register, and IBIEN = 1 and DAVAL = 1 in the I3C_DEVRO0.register, and the IBI is not yet acknowledged by the controller neither disabled via DISEC), else 0

The returned 2-byte STATUS[15:0] with format 2 on the I3C bus is then as follows:

• STATUS[15:8] = 0000 0000 (unused)
• STATUS[7:2] = 0000 (unused)
• STATUS[1] = 1 if a received DEFTGTS or a received DEFRGPA CCC is still under software processing, and the related event is not yet cleared by software (DEFF = 1 or GRPF = 1 in the I3C_EVR register); the controller must wait before issuing a GETACCR CCC (else it is not acknowledged)
• STATUS[0] = 1 if a DEFTGTS or DEFRGPA CCC may have been missed. This bit is asserted if a missed start is detected (WKPF = 1 in the I3C_EVR register), de-asserted if DEFF = 1 or GRPF = 1 in the I3C_EVR register.

Completion of a GETSTATUS CCC of format 1 is reported by STAF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if STAIE = 1 in the I3C_IER register).

Completion of a GETSTATUS CCC of format 2 is reported by GETF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if GETIE = 1 in the I3C_IER register).
49.9.10 I3C private read/write transfer, as controller

Figure 661 illustrates private read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

Figure 661. I3C private read/write messages, as controller

49.9.11 I3C private read/write transfer, as target

Figure 662 illustrates I3C private read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.
**Figure 662. I3C private read/write messages, as target**

**I3C private write** (when IP is acting as target)

- **S**: Start
- **Sr**: Start of previous private or I2C transfer
- **ACK**: Acknowledge
- **T**: Transition bit

1. Controller (& target) drive(s) SDA low/high Z in open-drain (arbitrable header)
2. Controller drives SDA in open-drain
3. Controller drives SDA in push-pull
4. Target drives SDA in push-pull
5. Previous private or I2C transfer

---

**Legend**

- **Controller (A & target) drive(s) SDA low/high Z in open-drain (arbitrable header):**
- **Controller drives SDA in open-drain:**
- **Controller drives SDA in push-pull:**
- **Target drives SDA in push-pull:**
- **Transition bit (parity bit for write data) from controller (drives SDA in push-pull):**
- **Transition bit (end of data for read data) from controller and/or target (drive SDA low/high Z in open-drain):**
- **Acknowledgments from the addressed target(s) (drive(s) SDA low in open-drai) with hand-off (switch back SDA control to controller):**
- **Acknowledgments from the addressed target(s) (drive(s) SDA low in open-drain) without hand-off (continue driving SDA):**

*(T): Target is not able to acknowledge an addressed private read/write if it is emitted without arbitrable header (i.e., after a start) and if it occurs at the same SCL clock cycle as an IBI request from this target. In order to avoid such an address-based arbitration collision, the controller should emit a private read/write with arbitrable header when the IBI request is enabled.
49.9.12 Legacy I²C read/write transfer, as controller

Figure 663 illustrates legacy I²C read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

Figure 663. Legacy I²C read/write messages, as controller
49.9.13 **I3C IBI transfer, as controller/target**

*Figure 664 illustrates IBI (in-band interrupt) transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.***

![Diagram of I3C IBI transfer, as controller/target](image-url)

### Legend

- **Controller drives SDA in push-pull**
- Target drives SDA in push-pull
- Target drives SDA in open-drain
- Target drives SDA in open-drain (start condition)
- Target drives SDA in open-drain (target initiates a start)
- Controller provides SCL clocking
- Controller drives SCL in push-pull
- Controller drives SCL in open-drain
- Controller drives SCL in push-pull (start condition)
- Controller drives SCL in open-drain (start condition)
- Controller drives SCL in push-pull (start condition)
- Controller drives SCL in open-drain (start condition)

(*) Target is not able to acknowledge an addressed private read/write if it is emitted without arbitrable header (i.e. after a start), and if it occurs at the same SCL clock cycle as an IBI request from this target. To avoid such an addressed arbitration collision, the controller should send a private read/write with arbitrable header when the IBI request is emitted.

(**) Repeated start (Sr) if there is a pending transfer to execute and if I3C_DEVRi.SUSP = 0, else stop (P) in push-pull.
When the peripheral acts as controller, the I3C_IBIDR register is used to receive the IBI data payload. Consequently, the IBI request from the target must not exceed a 4-byte data payload. If there is more information to be exchanged in the context of this in-band interrupt, the controller software must issue a private read.

### 49.9.14 I3C hot-join request transfer, as controller/target

*Figure 665* illustrates hot-join request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

**Figure 665. Hot-join request transfer, as controller/target**
49.9.15 I3C controller-role request transfer, as controller/target

Figure 666 illustrates controller-role request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 666. Controller-role request transfer, as controller/target

```
LEGEND

Controller drives SDA in push-pull
Target drives SDA low/high-Z in open-drain (and controller in open-drain high-Z)
Start request
Target initiates a start by driving SDA low in open-drain while SCL is high and waits for controller to pull SCL low

Controller drives SDA in open-drain (start condition)
CAS
Target provides SCL clocking after tSCL

Target is not able to acknowledge an addressed private read/write if it is emitted without arbitrable header (i.e. after a start) and if it occurs at the same SCL clock cycle as an IBI request from this target. In order to avoid such an address-based arbitration collision, the controller should emit a private read/write with arbitrable header when the IBI request is enabled.

Repeated start (Sr) if there is a pending transfer to execute, else stop (P)

(): Target is not able to acknowledge an addressed private read/write if it is emitted without arbitrable header (i.e. after a start) and if it occurs at the same SCL clock cycle as an IBI request from this target. In order to avoid such an address-based arbitration collision, the controller should emit a private read/write with arbitrable header when the IBI request is enabled.

(): Repeated start (Sr) if there is a pending transfer to execute, else stop (P)
```
49.10  I3C FIFOs management, as controller

49.10.1  C-FIFO management, as controller

When controller, as illustrated in figures of Section 49.9, C-FIFO can be used during any of the following transfers:

- broadcast CCC (*Figure 654, Figure 655, Figure 656*)
- direct read/write CCC (*Figure 656*)
  - command part, first message
  - data part, next message(s)
- private read/write (*Figure 661*)
- legacy I2C read/write (*Figure 663*)
- software-initiated error recovery (SCL forced to be stopped until next header message followed by HDR exit pattern)

*Figure 667* illustrates the management of the C-FIFO for queuing control word(s) on the I3C bus, when the I3C peripheral acts as controller.

---

1. This feature is implementation-dependent and can be unavailable. Refer to *Section 49.3.4.*
First, the software must initialize the C-FIFO management via CDMAEN in the I3C_CFGR register, to be written either:

- directly by the software (if CDMAEN = 0) at the control word level:
  - via polling mode (CFNFIE = 0 in the I3C_IER register): waiting for a next control word is requested by the hardware (CFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_CR register
  - via enabled interrupt notification if CFNFIE = 1
- by the allocated DMA channel (if CDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tc_dma):
  - as configured at block level, the DMA is automatically pushing-on/writing control word(s) into the I3C_CR register from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

In any case, if C-FIFO is empty and a restart must be emitted with a new control word, a C-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The DMA mode for the C-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1 in the I3C_EVR register), the C-FIFO is flushed automatically by the hardware.

49.10.2 TX-FIFO management, as controller

When controller, as shown in figures of Section 49.9, TX-FIFO can be used during any of the following transfers:

- broadcast or direct CCC (including ENTDAA and RSTACT), if a defining/sub-command byte or if data byte(s) are present (Figure 654, Figure 655, Figure 656)
- private write (Figure 661)
- legacy I2C write (Figure 663)

Figure 668 illustrates the management of the TX-FIFO for queuing data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as controller.
First, the software must initialize the TX-FIFO management via the following fields in the I3C_CFGR register:

- **TXDMAEN**: enable/disable DMA mode for TX-FIFO
- **TXTHRES**: push-on data byte(s) or word(s) into TX-FIFO
- **TMODE**: enable/disable TX-FIFO and C-FIFO preload

Then, depending upon bit TXDMAEN in the I3C_CFGR register, the TX-FIFO is written either:

- directly by the software (if TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon bit TXTHRES in the I3C_CFGR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (if TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon bit TXTHRES in the I3C_CFGR register) from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.
An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last data byte/word to be transmitted is flagged by TXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software to update the pointer to the buffer where the byte(s)/word(s) of the next message is/are stored.

When frame completion is reported (FCF = 1 in the I3C_EVR register, and the corresponding interrupt is enabled), the TX-FIFO is empty.

If the TX-FIFO is empty and a data byte must be transmitted on the I3C bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1), the TX-FIFO is automatically flushed by the hardware.

**No C-FIFO/TX-FIFO preload**

As defined in Table 516, C-FIFO size is two words, TX-FIFO size is 8 bytes.

When no C-FIFO/TX-FIFO preload is configured (TMODE = 0 in the I3C_CFGR register), the I3C peripheral emits a start on the I3C bus as soon as the first control word is written into the C-FIFO. Then, it decodes the I3C_CR register, and requires a next data byte/word to be written, if needed within this message. As soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus or if the C-FIFO gets available room), this control word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as another data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus, and if TX-FIFO is not full, and if data byte(s)/word(s) must be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

**C-FIFO and TX-FIFO preload**

When C-FIFO/TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), before emitting a start on the bus, the I3C peripheral waits for loading as much as possible both the C-FIFO and the TX-FIFO, as follows:

- wait for a first control word to be written into the C-FIFO
- wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the first control word (if RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size
- If TX-FIFO is not full and if the first control word is not the last of the frame (MEND = 0 in the I3C_CR register):
  - Wait for a second control word to be written into the C-FIFO, then C-FIFO is full.
  - If TX-FIFO is not full, wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the second control word (RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.

Then, as soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus), this control...
word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as a next data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus and if TX-FIFO is not full and if data byte(s)/word(s) are to be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

49.10.3 RX-FIFO management, as controller

When controller, as shown in figures of Section 49.9, RX-FIFO is used during any of the following transfers:

- broadcast ENTDAA CCC (Figure 655)
- direct CCC read (Figure 654), including direct RSTACT CCC read (Figure 656)
- private read (Figure 661)
- legacy I2C read (Figure 663)

Figure 669 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received from the I3C bus, when the I3C peripheral acts as controller.

Figure 669. RX-FIFO management, as controller
First, the software must initialize the RX-FIFO management via the following fields of the I3C_CFGR register:

- **RXDMAEN**: enable/disable DMA mode for RX-FIFO
- **RXTHRES**: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR register, depending upon bit RXTHRES in the I3C_CFG register
  - via enabled interrupt notification if RXFNEIE = 1 in the I3C_IER register
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA automatically pops-out/reads data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon bit RXTHRES), and writes them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last received data byte/word from the I3C bus is flagged by IRXLASTF = 1 in the I3C_EV register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software for updating the pointer to the buffer where is/are stored the data byte(s)/word(s) of the next message.

If RX-FIFO is full and a data byte is received on the I3C bus, an RX-FIFO overrun is reported (ERRF = 1 in the I3C_EV register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

**Early read termination from the target**

A private read message can be early completed (also known as prematurely ended) by the addressed target.

- **If RXDMAEN = 1:**
  - the software must allocate, for the DMA request i3c_rx_dma, a DMA channel x, capable of peripheral early termination (refer to DMA implementation section).
  - The software must configure the DMA channel x to enable the DMA peripheral-flow control mode via PFREQ = 1 in the DMA_CxTR2 register, to be able to perform a(n) (early or not) DMA block completion.
  - Then, on block completion, the software can read BR1.BNDT[15:0] in DMA_Cx and/or DMA_CxSAR register(s), to get the effective number of DMA transferred bytes.
- **If RXDMAEN = 0:**
  - at message level, the last received data byte/word from the I3C bus is flagged by RXFNEF = 1 and RXLASTF = 1 in the I3C_EV register.
In any case, if the S-FIFO is disabled (if SMODE = 0 in the I3C_CFGR register), the software is notified that an early read termination occurs by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled. Then, software can read the status register I3C_SR to check information related to the last message, and get the number of received data bytes on the prematurely ended read transfer (XDCNT[15:0] in the I3C_SR register).

In any case, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the software or the DMA (depending upon SDMAEN in the I3C_CFGR register) must read for each message the status register I3C_SR. The number of effective received data bytes on the prematurely ended read message is reported by XDCNT[15:0] (and then ABT = 1) in the I3C_SR register.

For more information, refer to I3C status register (I3C_SR) and Section 49.10.4.

49.10.4 S-FIFO management, as controller

When controller, S-FIFO can be used by the software to be able to read the I3C status register (I3C_SR) for each transferred message.

Reading status register with disabled S-FIFO

If SMODE = 0 in the I3C_CFGR register, the S-FIFO is disabled and the status register can be read as a usual register:

• the register content is overwritten by hardware when is transforee a new message
• I3C_SR contains the status of the last transferred message
• SCL clock is not stalled if status register is not read.

If SMODE = 0, for the specific case of a private read prematurely ended by the target:

• the software is notified by the flag RXTGTENDF = 1 in the I3C_EVR register and the corresponding interrupt if enabled.
• Then, until that the software has cleared the event flag write and set CRXTGTENDF = 1 in the I3C_CEVR register:
  – no more data byte can be received on the I3C bus and written by the hardware into I3C_RDR/I3C_RDWR registers
  – I3C_SR cannot be updated
  – SCL clock is stalled if needed

Typically, I3C_SR can be read when FCF = 1, or ERRF = 1, or RXTGTENDF = 1 in the I3C_EVR register. XDCNT[15:0] in the I3C_SR register can be read to get the effective number of received data bytes on a private read, after an early termination from the target (refer to Section 49.10.3).

Reading status register with enabled S-FIFO

If SMODE = 1 in the I3C_CFGR register, the S-FIFO is enabled. Figure 670 illustrates the management of the S-FIFO for queuing and popping-out a status word for each executed message on the I3C bus, when the I3C peripheral acts as controller.
First, the software must initialize the S-FIFO management via the SDMAEN field (enable/disable DMA mode for S-FIFO) in the I3C_CFGR register. Then, depending on SDMAEN, the S-FIFO is read either:

- directly by the software (if SDMAEN = 0):
  - via polling mode (SFNEIE = 0 in the I3C_IER register): waiting for a next status word is requested by the hardware (SFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_SR register
  - via enabled interrupt notification (SFNEIE = 1)
- by the allocated DMA channel (if SDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rs_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading status words from the I3C_SR register and writing them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

Each message status must be read, else an overrun error is asserted by the hardware (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register) when the S-FIFO is full and a next message status must be written. The corresponding interrupt is raised if enabled by ERRIE = 1 in the I3C_IER register.

The frame completion (FCF = 1 in the I3C_EVR register) is reported only when the S-FIFO is empty.
The configuration for the S-FIFO management can be modified when the I3C peripheral is not in active state.

49.11 I3C FIFOs management, as target

49.11.1 RX-FIFO management, as target

When target, as shown in figures of Section 49.9, the RX-FIFO is used during any of the following received and acknowledged transfers:
- broadcast DEFTGTS CCC (Figure 659)
- broadcast DEFGRPA CCC (Figure 660)
- private write (Figure 662)

Figure 671 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received from the I3C bus, when the I3C peripheral acts as target.
First, the software must initialize the RX-FIFO management via the following I3C_CFRG register fields:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (if RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR registers, depending upon RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon RXTHRES) and writing them into its memory destination buffer, until the transfer completion (in the I3C_EVR register, FCF = 1 in case of private write, or GRPF = 1 in case of a DEFFRPA CCC, or DEFF = 1 in case of DEFTGTS CCC), unless a transfer error occurs.

If RX-FIFO is full and a new data byte is received on the I3C bus, a RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

When the transfer is completed (in the I3C_EVR register, FCF = 1, or GRPF = 1, or DEFF = 1):

- RX-FIFO is empty
- Until software has not processed the RX data buffer corresponding to the completed private write / DEFTGTS / DEFFRPA transfer (meaning until software has not cleared the corresponding flag write and set CFCF / CDEFF / CGRPF to 1 in the I3C_CEVR register), if a next private write / DEFTGTS CCC / DEFFRPA CCC is received and a data byte must be written by the hardware into the RX-FIFO, a transfer error is reported with a RX-FIFO overrun (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

### 49.11.2 TX-FIFO management, as target

#### Main scheme

When target, as shown in figures of Section 49.9, the TX-FIFO is used only during a private read (Figure 662).

Figure 672 illustrates the management of the TX-FIFO for queuing and pushing-on data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as target.
First, the software must initialize the TX-FIFO management and write I3C_CFGR with

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) to TX-FIFO

Then, before receiving a private read on the I3C bus, the software must configure the I3C target transmit configuration register (I3C_TGTTDR) to preload the TX-FIFO with a number of data bytes (write TGTTDCNT[15:0] and PRELOAD = 1 in a single access), so that data bytes from the target are ready to be transmitted on the I3C bus:

- If PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size, TX-FIFO is first preloaded up to the FIFO size.
- If PRELOAD = 1 and TGTTDCNT[15:0] ≤ TX-FIFO size, TX-FIFO is preloaded up to TGTTDCNT[15:0].

Note: TX-FIFO size is 8 bytes (refer to Table 516).
Depending upon TXDMAEN, the TX-FIFO is preloaded either:

- directly by the software (TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon TXTHRES in the I3C_CFGFR register
  - via enabled interrupt notification if TXFNFIE = 1

- by the allocated DMA channel (TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-onwriting data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon TXTHRES) from its memory source buffer, until the transfer completion (FCF = 1 in the IEC_EVR register), unless a transfer error occurs.

Then, in the same way, either directly by the software or by the allocated DMA channel, if there are remaining TGTTDCNT[15:0] in the I3C_TGTTDR register to be loaded into TX-FIFO for continuing the private read (set PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size), and provided that the private read is not yet completed by the controller:

- If TXTHRES = 0: when a byte is transmitted on the I3C bus, a next byte is preloaded into TX-FIFO
- If TXTHRES = 1: when four bytes are transmitted on the I3C bus, the next word is preloaded into TX-FIFO.

The private read transfer is completed (FCF = 1 in the I3C_EVR register) when either the target or the controller first terminates the data byte transfer.

On transfer completion, the software can:

- read XDCNT[15:0] in the I3C_SR register: the effective number of transmitted data bytes
- read TGTTDCNT[15:0] in the I3C_TGTTDR register: the remaining number of bytes to be loaded and transmitted on the I3C bus
- flush TX-FIFO: write and set (or not) TXFLUSH = 1 in the I3C_CFGFR register, continue (or not) for another private read, depending upon the user application

If TX-FIFO is empty and a data byte must be transmitted on the I3C bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_EVR register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

**Alternative without I3C_TGTTDR, if less bytes than the TX-FIFO size**

Alternatively to the use of the I3C_TGTTDR register, as shown in the Figure 673, when the DMA is not used (TDMAEN = 0), provided that the number of data bytes to be read on the I3C bus is less that the TX-FIFO size, the software can directly prepare and fill the TX-FIFO with the number of bytes to be read directly by the software, by successive writes to the I3C_TDR or I3C_I3C_TDWR, depending upon TXTHRES.
49.12 I3C error management

49.12.1 Controller error management

Table 525 enumerates the I3C bus error conditions, and, for each of them, the corresponding detection, action and reporting when the I3C peripheral acts as controller.

<table>
<thead>
<tr>
<th>Error type</th>
<th>Description</th>
<th>Error detection</th>
<th>Controller action</th>
<th>Reported error(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE0</td>
<td>Illegally formatted CCC (e.g. less returned data byte(s))</td>
<td>Prematurely ended read data by target on a direct CCC read(2)</td>
<td>Hardware emits a stop.</td>
<td>ERRF = 1 and PERR = 1 and CODERR[3:0] = 0000</td>
</tr>
</tbody>
</table>
### Table 525. I3C controller error management (continued)

<table>
<thead>
<tr>
<th>Error type</th>
<th>Description</th>
<th>Error detection</th>
<th>Controller action</th>
<th>Reported error(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE1</td>
<td>Monitoring error</td>
<td>An incorrect ACK is detected at the end of a legacy I2C read</td>
<td>Hardware keeps SCL running for nine clock cycles and another byte to be possibly exchanged, then emits again a NACK, followed by a stop.</td>
<td>ERRF = 1 and PERR = 1 and CODERR[3:0] = 0001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Restart or stop cannot be generated at the end of an I3C SDR read</td>
<td>SCL is kept running. Software can stop SCL by writing a control word with MTYPE[3:0] = 0000 in the I3C_CR register (header message), then must typically wait at least 150 µs before emitting another message.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>After a CE1 error, a start cannot be generated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CE2</td>
<td>No response to broadcast address (0b111_1110)</td>
<td>Header (0b111_1110 + RnW = 0) is detected as NACK-ed during a message different from escalation fault or reset pattern message</td>
<td>Hardware emits an HDR exit pattern followed by a stop.</td>
<td>ERRF = 1 and PERR = 1 and CODERR[3:0] = 0010</td>
</tr>
<tr>
<td>CE3</td>
<td>Failed controller hand-off</td>
<td>New controller has not driven SCL low after having SDA low (after a start via a test header or a start request from a target) and after the delay time defined by its activity state has elapsed</td>
<td>Hardware emits a start + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.</td>
<td>ERRF = 1 and PERR = 1 and CODERR[3:0] = 0011</td>
</tr>
<tr>
<td>-</td>
<td>Incorrect returned 7-bit target address with parity bit on a GETACCCR CCC</td>
<td>Incorrect dynamic address and/or parity bit is detected</td>
<td>Hardware cancels the GETACCCR CCC by emitting a Restart + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.</td>
<td>ERRF = 1 and DERR = 1</td>
</tr>
<tr>
<td>-</td>
<td>Addressed target is NACK-ed on a direct CCC read for the first time</td>
<td>NACK is detected</td>
<td>Hardware performs a single-retry by emitting a Restart + 7-bit same target address + RnW = 1.</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>Addressed target is NACK-ed on either a direct CCC write, an I3C private read/write, a legacy I2C, or a direct CCC read for the second time</td>
<td>NACK is detected</td>
<td>Hardware emits a stop.</td>
<td>ERRF = 1 and ANACK= 1</td>
</tr>
</tbody>
</table>
Table 525. I3C controller error management (continued)

<table>
<thead>
<tr>
<th>Error type</th>
<th>Description</th>
<th>Error detection</th>
<th>Controller action</th>
<th>Reported error(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the first time</td>
<td>NACK is detected</td>
<td>Hardware performs a single-retry assignment loop by emitting a Restart + 0b111_1110 + RnW = 1, followed by an ACK and 8-byte read data from the (most-priority) target, followed by the assigned address + parity bit.</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the second time</td>
<td>NACK is detected</td>
<td>Hardware emits a stop.</td>
<td>ERRF = 1 and DNACK= 1</td>
</tr>
<tr>
<td>-</td>
<td>Write data is NACK-ed on a legacy I2C write</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>Control word, status word, transmitted data or read data is not written/read in time vs. I3C bus timings</td>
<td>SCL stall timeout</td>
<td>Hardware emits a stop.</td>
<td>ERRF = 1 and (COVR = 1 or DOVR = 1)</td>
</tr>
</tbody>
</table>

1. ERRF in the I3C_EVR register, PERR, CODERR[3:0], DERR, ANACK, DNACK, COVR, and DOVR in the I3C_SER register.
2. MIPI v1.1: on a GETCAPS CCC, the number of received data bytes can be 2, 3 or 4. However a target compliant with MIPI v1.0 can return only the first byte as per previously named GETHDCRP CAP CCC. As a result, CE0 is not generated if the number is lower than 4. On a GETMXDS CCC, the number of received data bytes can be 2 or 5. CE0 is not generated if the number is lower than 5.

49.12.2 Target error management

Table 526 enumerates the I3C bus error conditions, the corresponding detection, action, and reporting when the I3C peripheral acts as target.

Table 526. I3C target error management

<table>
<thead>
<tr>
<th>Error type</th>
<th>Description</th>
<th>Error detection</th>
<th>Next (when emitted by the controller)</th>
<th>Reported error(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TE0</td>
<td>Invalid broadcast address (0b111_1110+RnW = 0) or Invalid 7-bit dynamic address + RnW = 1 after DAA assignment</td>
<td>A forbidden address is detected after a start or a repeated start</td>
<td>Hardware waits for an HDR exit pattern</td>
<td>ERRF = 1 and PERR = 1 and ODERR[3:0] = 1000</td>
</tr>
<tr>
<td>TE1</td>
<td>CCC code</td>
<td>A CCC code is detected with a parity error</td>
<td></td>
<td>ERRF = 1 and PERR = 1 and ODERR[3:0] = 1001</td>
</tr>
</tbody>
</table>
49.13  I3C wake-up from low-power mode(s)

49.13.1  Wake-up from Stop

The user must first configure the reset and clock controller (RCC) to set up the clock data path down to the I3C peripheral: to select the source oscillator for the I3C kernel clock, the source oscillator for the I3C APB clock, and to set the clocks frequency. At initialization via the RCC, the user must enable the I3C clocks for a given I3C peripheral to be functional, separately for Run/Sleep mode and for Stop mode. For more details about RCC programming, refer to Section 11: Reset and clock control (RCC).

The I3C hardware automatically manages its own clocks gating and generates a separated clock request output signal to the RCC for its kernel clock and its APB clock, whenever the device is in Run, Sleep or Stop mode.
When entering a Stop mode, the \( V_{\text{CORE}} \) domain is supplied, by default any clock oscillator is disabled, and in any case, neither the system clock nor a peripheral clock is running in the domain.

**As controller: wake-up on an IBI without MDB, a hot-join request or a controller-role request**

When the peripheral acts as controller, before the product enters a low-power mode, the software must issue an ENTAS\(x \) CCC, generally with \( x = 0, 1, 2 \) or \( 3 \), to inform targets that the I3C controller is not expected to exit from idle state neither to communicate on the I3C bus before an interval of, respectively, \( 1 \ \mu s \), \( 100 \ \mu s \), \( 2 \) ms or \( 50 \) ms, has elapsed. This delay defines the \( T_{\text{CAS}} \) delay for the controller to set the SCL bus clock low and running, after a start condition. More specifically for a Stop mode, the CCC must be restricted to an ENTAS\(x \) with the value \( x=1, 2, \) or \( 3 \).

The general scheme for the I3C wake-up from Stop is the following:

1. First the I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high).
   - As controller, as initiated by an external I3C target device for a hot-join/in-band interrupt/controller-role request. Once the kernel clock is provided and running, the I3C hardware uses an internal timer to wait for the corresponding \( T_{\text{CAS}} \) time to elapse, then drives low SCL and continues toggling SCL to let the target perform its hot-join/in-band interrupt/controller-role request on the I3C bus.

2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds from HSI). The user may keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption.
   - The I3C peripheral maintains the kernel clock request until the generation of the Stop condition on the I3C bus.

3. After that the kernel clock is running, as controller the I3C peripheral requests its APB clock to the system:
   - On the ACKed address of a received IBI request (Figure 664), and it maintains the APB clock request until that IBIF is cleared.
     
     If the IBI is without MDB: a Stop is normally generated on the I3C bus, even if the APB clock is not yet provided.
   - On the ACKed address of a received controller-role request (Figure 665), and it maintains the APB clock request until that CRF is cleared.
   - On the ACKed address of a hot-join request (Figure 666), and it maintains the APB clock request until that HJF is cleared.

4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken.

5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.

6. The system enables the system clock that drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after a delay.

7. With the APB clock running, the peripheral can log the I3C transfer in its status and data registers. When the bus transfer is completed, the peripheral generates the corresponding flag (IBIF/CRF/HJF), and the enabled interrupt can wake up the CPU.

**As target: wake-up on a reset pattern**
The target reset pattern is a specific scheme for a controller to wake up and possibly reset a target from a low power mode. It consists both in the RSTACT CCC and in the in-band reset pattern generation.

As target, the sequence for the I3C wake-up from Stop on a reset pattern is the following:
1. When the peripheral detects a reset pattern on the bus (14 SDA transitions while SCL is kept low), it requests its APB clock to the system to set the RSTF flag of the I3C_EVR register.
2. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
3. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
4. The system enables the source oscillator of the system clock, which gets ready after few microseconds.
5. With the APB clock running, the I3C peripheral can raise the RSTF flag of the I3C_EVR register, and the enabled interrupt can wake up the CPU.

As target: wake-up on a missed start
1. The peripheral requests its kernel clock to the system:
   - On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high)
   - As target, as initiated by an external I3C device, whatever controller or target.
   - If the kernel clock is not provided before that SCL is driven low by the external controller, then the I3C target detects that a I3C bus start is missed and waits for the kernel clock to be provided. It can be noted that an I3C controller message intended to be addressed to it may have been missed (and NACKed). Or may have been missed any I3C CCC or private message from the controller to another addressed target or an IBI/CR/HJF start request from another target.
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds if HSI). The user may keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption. Especially if the controller is in an Activity State of x = 0 (with a 1 µs T_{CAS} latency).
3. After that the kernel clock is running, as target, the I3C peripheral requests its APB clock to the system to raise the WKPF flag conditioned by non-user text or:
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock, which drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after few microseconds.
7. With the APB clock running:
   - The missed start flag (WKPF of the I3C_EVR register) is raised conditioned by non-user text: if it occurred, and the enabled interrupt can wake up the CPU. It is then known that an I3C bus transaction was missed, but not whether the target was addressed or not. The software should use a timeout in order to return to Stop mode, if it is not addressed by the controller again within this interval.
## 49.14 I3C in low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. I3C interrupts cause the device to exit Sleep mode.</td>
</tr>
<tr>
<td>Stop(^{(1)})</td>
<td>The content of the I3C registers is kept when entering Stop mode. I3C hardware manages automatically its own clocks gating and generates, for its kernel clock and APB clock, a clock request output signal to the RCC. I3C interrupts can cause the device to wake up and exit Stop mode. I3C transfers can occur and can be assisted with an autonomous DMA for data transfers to/from memory, provided that the autonomous DMA can operate in Stop mode. Refer to Section 49.13 for more details.</td>
</tr>
<tr>
<td>Standby</td>
<td>The I3C peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

1. Refer to Table 515 to know if Stop mode is supported, and which one.
# I3C interrupts

## Table 528. I3C interrupt requests

<table>
<thead>
<tr>
<th>Interrupt Acronym</th>
<th>Interrupt Event</th>
<th>Used as Interrupt enable: field in I3C_IER</th>
<th>Event Flag: field in I3C_EV</th>
<th>Event Clear Method: write 1 to field in I3C_CEVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVT</td>
<td>A control word is requested</td>
<td>X - CFNFIE CFNFF -</td>
<td>X X RXFNFIE RXFNFF -</td>
<td>X X FCIE FCF CFCF</td>
</tr>
<tr>
<td>EVT</td>
<td>A status word is available</td>
<td>X - SFNEIE SFNEF -</td>
<td>X X TXFNFIE TXFNFF -</td>
<td>X X FCIE FCF CFCF</td>
</tr>
<tr>
<td>EVT</td>
<td>Data to be transmitted are requested</td>
<td>X X TXFNFIE TXFNFF -</td>
<td>X X FCIE FCF CFCF</td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>Received data are available</td>
<td>X X RXFNEIE RXFNEF -</td>
<td>X X FCIE FCF CFCF</td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>Controller: frame transfer is completed</td>
<td>X X FCIE FCF CFCF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>A private read transfer is prematurely ended by the target (and SMODE = 0 in the I3C_CFGR register)</td>
<td>X - RXTGTENDIE RXTGTENDF CRXTGTENDF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>An IBI request is received</td>
<td>X - IBIIE IBIIF CIBIF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>A controller-role request is received</td>
<td>X - CRIE CRF CCRF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>A hot-join request is received</td>
<td>X - HJIE HJF CUF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>IBI request is completed</td>
<td>- X IBIENDIE IBIENDF CIBIENDF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>I3C bus start is missed</td>
<td>- X WKPIE WKPF CWKPF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>Direct GETACCR CCC is received</td>
<td>- X CRUPDIE CRUPDF CCRUPDF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>Direct GETSTATUS CCC is received</td>
<td>- X STAIE STAF CSTAF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>Any direct GETxxx CCC (except GETSTATUS) is received</td>
<td>- X GETIE GETF CGETF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>Dynamic address is updated (broadcast ENTDA or RSTDAA, or direct SETNEWDA is received)</td>
<td>- X DAUPDIE DAUPDF CDAUPDF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>Direct SETMWL CCC is received</td>
<td>- X MWLUPDIE MWLUPDF CMWLUPDF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>Direct SETMRL CCC is received</td>
<td>- X MRLUPDIE MRLUPDF CMRLUPDF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>A reset pattern is detected</td>
<td>- X RSTIE RSTF CRSTF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>Bus activity state is updated (direct/broadcast ENTASx CCC is received)</td>
<td>- X ASUPDIE ASUPDF CASUPDF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>Broadcast/direct ENEC/DISEC CCC is received</td>
<td>- X INTUPDIE INTUPDF CINTUPDF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>Broadcast DEFTGTS CCC is received</td>
<td>- X DEFIE DEFF CDEFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>Broadcast DEFGRPA CCC is received</td>
<td>- X GRIE GRPF CGRPF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVT</td>
<td>An error occurred</td>
<td>X X ERRIE ERRF CERRF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
49.16 **I3C registers**

The I3C registers must be accessed with a 32-bit word aligned address.

**Note:** I3C_RDR and I3C_TDR registers must be accessed with a single significant LSB data byte for, respectively, reading RX-FIFO and writing TX-FIFO.

### 49.16.1 I3C message control register (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This register must be used to control the message to emit on the I3C bus:

- when I3C acts as controller (bit[30] = MTYPE[3] = 0): if there is no CCC code to be emitted bits[29:27] = MTYPE[2:0] differ from 110; else the alternate register description Section 49.16.2 must be considered.

When I3C acts as controller:

- If the control FIFO (C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by software, or automatically by DMA, as defined by CDMAEN in the I3C_CFGR register.
- If C-FIFO is empty and a restart must be emitted with a new control word, the I3C hardware asserts the control FIFO error underrun flag (COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.
- After the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

When I3C acts as target, this register is used in register mode:

- Software writes into this register to initiate a command (IBI, controller-role or hot-join request) on the I3C bus.
- C-FIFO is disabled, and there is no DMA mode neither for control words.

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>30</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>29</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**DCNT[15:0]**

```
<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Bit 31 **MEND:** Message end type/last message of a frame (when the I3C acts as controller)

- 0: this message from controller is followed by a repeated start (Sr), before another message must be emitted
- 1: this message from controller ends with a stop (P), being the last message of a frame
Bits 30:27 MTYPE[3:0]: Message type (whatever I3C acts as controller/target)

Condition: when I3C acts as I3C controller

0000: SCL clock is forced to stop until a next control word is executed

   Bits [26:0] are ignored. On a CE1 error detection (ERRF = 1 in the I3C_EVR register and CODERR[3:0] = 0001 in the I3C_SER register) where a start/restart/stop is prevented from being generated, the software must use this message type for SCL "stuck at" recovery. Refer to Table 525.

0001: header message

   Bits [26:0] are ignored. If the addressed target is not responding with an ACK to a private/direct message, as an escalation stage after a failed GETSTATUS tentative, the software must program this with EXITPTRN = 1 in the I3C_CFGR register, so that an HDR exit pattern is emitted on the bus, whatever the header is ACK-ed or NACK-ed (to avoid the target to consider that the I3C bus is in HDR mode). Refer to Table 525 and MIPI specification about escalation handling.

0010: private message (refer to Figure 661)

   Bits [23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
   Bit[16] (RNW) is the emitted RnW bit.
   Bits [15:0] (DCNT[15:0]) are the number of programmed data bytes.
   The transferred private message is:
   - \( \{S + 0b111_1110 + RnW = 0 + Sr/Sr*\} + 7\text{-bit DynAddr} + RnW + (8\text{-bit Data} + T)^* + Sr/P. \)
   - After an S (start), depending upon bit NOARBH in the I3C_CFGR register, the arbitrable header \( (0b111_1110 + RnW = 0) \) is inserted or not.
   - \( Sr* \) : after an Sr (repeated start), the hardware automatically inserts \( (0b111_1110 + RnW = 0) \) if needed, if it follows a previous message without ending by a P (stop).

0011: direct message (second part of an I3C SDR direct CCC command) (refer to Figure 654)

   Bits [23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
   Bit[16] (RNW) is the emitted RnW bit.
   Bits [15:0] (DCNT[15:0]) are the number of programmed data bytes.
   The transferred direct message is: \( Sr + 7\text{-bit DynAddr} + RnW + (8\text{-bit Data} + T)^* + Sr/P \)

0100: legacy I2C message (refer to Figure 663)

   Bits [23:17] (ADD[6:0]) are the emitted 7-bit static address.
   Bit[16] (RNW) is the emitted RnW bit.
   Bits [15:0] (DCNT[15:0]) are the number of programmed data bytes.
   The transferred legacy I2C message is:
   - \( \{S + 0b111_1110 + RnW = 0 + Sr/Sr*\} + 7\text{-bit StaAddr} + RnW + (8\text{-bit data} + T)^* + Sr/P. \)
   - After an S, depending on NOARBH, the arbitrable header \( (0b111_1110 + RnW = 0) \) is inserted or not.
   - \( Sr* \) : after an Sr (repeated start), the hardware automatically inserts \( (0b111_1110 + RnW = 0) \) if needed (if it follows a previous message without ending by a P (stop)).

Others: reserved

Condition: when I3C acts as I3C target

1000: hot-join request (W) (refer to Figure 665)

   The transferred hot-join request is \( \{S + 0b000_0010 \text{ addr} + RnW = 0 \) .
   Writing the control word initiates the hot-join request if target is allowed to do so (HJEN = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1001: controller-role request (W) (refer to Figure 666)

   The transferred controller-role request is \( \{S + DA[6:0] + RnW = 0 \) (DA in the I3C_DEVR0 register).
   Writing the control word initiates the controller-role request if target is allowed to do so (CREN = 1 and DAVAL = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1010: IBI (in-band interrupt) request (R) (refer to Figure 664)

   Bits [15:0] (DCNT[15:0]) are the number of the IBI data payload (including the first MDB), if any.
   The transferred IBI request is \( \{S + DA[6:0] + RnW = 1 \) + optional IBI data payload.
   Writing the control word initiates the IBI request if target is allowed to do so (IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.
   When acknowledged from controller, the transmitted IBI payload data (optional, depending upon BCR2 in the I3C_BCR register) is defined by DCNT[15:0] in the I3C_CR register and I3C_IBIDR, and must be consistently programmed vs. the IBI payload data size defined by IBIP[2:0] in the I3C_IBIDR register.

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.
49.16.2 I3C message control register [alternate] (I3C_CR)

Address offset: 0x000
Reset value: 0x0000 0000

This write register description must be used to control the message for when the controller has to emit a CCC (whatever is the type of the CCC: for a CCC broadcast, a CCC direct, or a CCC Enter HDR).

This is the alternate description of register I3C_CR, for when MTYPE[3:0] = 0110. Else refer to Section 49.16.1.

If the control FIFO (also known as C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by the software or automatically by DMA, as defined by the CDMAEN bit in the I3C_CFGR register.

When the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.
Bit 31 **MEND**: Message end type/last message of a frame (when I3C acts as controller)

- 0: this message from controller is followed by a repeated start (Sr), before another message must be emitted
- 1: the message from the controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (when I3C acts as controller)

**Condition: when I3C acts as I3C controller**

- **0110**: broadcast/direct CCC command (refer to Table 524, Figure 654, Figure 655, Figure 656)
  - Bits[23:16] (CCC[7:0]) are the emitted 8-bit CCC code
  - Bits[15:0] (DCNT[15:0]) are the number of the CCC defining bytes, or CCC sub-command bytes, or CCC data bytes.
  - If Bit[23] = CCC[7] = 1: this is the first part of an I3C SDR direct CCC command
  - The transferred direct CCC command (first part) message is:
    - {S / S + 0b111_1110 + RnW = 0 / Sr+*} + (direct CCC + T) + (8-bit Data + T)* + Sr
    - After an S (start), depending upon NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
    - Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).
  - If Bit[23] = CCC[7] = 0: this is an I3C SDR broadcast CCC command (including ENTDAA).
  - The transferred broadcast CCC command message is:
    - {S / S + 0b111_1110 + RnW = 0 / Sr+*} + (broadcast CCC + T) + (8-bit Data + T)* + Sr/P
    - After an S (start), depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
    - Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

**Others**: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:16 **CCC[7:0]**: 8-bit CCC code (when I3C acts as controller)

- If bit[23] = CCC[7] = 1, this is the first part of an I3C SDR direct CCC command.
- If bit[23] = CCC[7] = 0, this is an I3C SDR broadcast CCC command (including ENTDAA).

Bits 15:0 **DCNT[15:0]**: Count of related data to the CCC command to transfer as CCC defining bytes, or CCC sub-command bytes, or CCC data bytes, in bytes

- Linear encoding up to 64 Kbytes - 1.
- 0x0000: no data to transfer.
- Note: Value mandatory when emitting ENTDAA broadcast CCC (refer to Figure 655).
- 0x0001: 1 byte
- Note: Value mandatory when emitting RSTACT direct/broadcast CCC (refer to Figure 656).
- 0x0002: 2 bytes
- ...
- 0xFFFF: 64 Kbytes - 1 byte
49.16.3 I3C configuration register (I3C_CFGR)

Address offset: 0x004
Reset value: 0x0000 0000

This register is used to configure:
- features that apply when the I3C acts as controller or target: RX-FIFO and TX-FIFO management (RXDMAEN, RXTHRES, RXFLUSH, TXDMAEN, TXTHRES, TXFLUSH), I3C peripheral role (CRINIT)
- dedicated features when the I3C acts as a controller: frame-based control-word triggering (TSFSET), FIFOs management (TMODE, SMODE, SFLUSH, SDMAEN, CDMAEN), and miscellaneous ones (HJACK, HKSDAEN, EXITPTRN, RSTPATRN, NOARBH)

The configuration fields CRINIT, HKSDAEN can be modified only when EN = 0. This condition is respected if they are modified at the same time when EN is set to 1 (it is not necessary to set EN later on, with another write operation).

| Bit 31 | Reserved, must be kept at reset value. |
| Bit 30 | TSFSET: Frame transfer set (software trigger) (when I3C acts as controller) |
| This bit can only be written. When I3C acts as I3C controller: |
| 0: no action |
| 1: setting this bit initiates a frame transfer by causing the hardware to assert the flag CFNFF in the I3C_EVR register (C-FIFO not full and a control word is needed) |
| Note: If this bit is not set, the other alternative for the software to initiate a frame transfer is to directly write the first control word register (I3C_CR) while C-FIFO is empty (CFEF = 1 in the I3C_EVR register). Then, if the first written control word is not tagged as a message end (MEND = 0 in the I3C_CR register), it causes the hardware to assert CFNFF. |
| Bits 29:24 | Reserved, must be kept at reset value. |
| Bit 23 | Reserved, must be kept at reset value. |
| Bit 22 | Reserved, must be kept at reset value. |
| Bit 21 | CFLUSH: C-FIFO flush (when I3C acts as controller) |
| This bit can only be written. |
| 0: no action |
| 1: flush C-FIFO |
Bit 20  **CDMAEN**: C-FIFO DMA request enable (when I3C acts as controller)
When I3C acts as controller:
0: DMA mode is disabled for C-FIFO
   - Software writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame
   - A next control word transfer can be written by software either via polling on the flag CFNFF = 1 in the I3C_EVR register, or via interrupt notification (enabled by CFNFIE = 1 in the I3C_IER register).
1: DMA mode is enabled for C-FIFO
   - DMA writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame.
   - A next control word transfer is automatically written by the programmed hardware (via the asserted C-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 19  **TMODE**: Transmit mode (when I3C acts as controller)
When I3C acts as controller, this bit is used for the C-FIFO and TX-FIFO management vs. the emitted frame on the I3C bus.
0: C-FIFO and TX-FIFO are not preloaded before starting to emit a frame transfer.
   A frame transfer starts as soon as the first control word is present in C-FIFO.
1: C-FIFO and TX-FIFO are first preloaded (also TX-FIFO if needed, depending on the frame format) before starting to emit a frame transfer. Refer to Section 49.10.2 for more details.

Bit 18  **SMODE**: S-FIFO enable / status receive mode (when I3C acts as controller)
When I3C acts as controller, this bit is used to enable the FIFO for the status (S-FIFO) of the exchanged message on the I3C bus.
When I3C acts as target, this bit must be cleared.
0: S-FIFO is disabled
   - Status register (I3C_SR) is used without FIFO mechanism.
   - There is no SCL stalling if a new status register content is not read.
   - Status register must be read before being overwritten by the hardware.
   - Must have SDMAEN = 0 in the I3C_CFGR register.
1: S-FIFO is enabled.
   - Each message status must be read.
   - There is SCL stalling when the S-FIFO is full and a next message status must be read.
   - S-FIFO overrun error is reported after the maximum SCL clock stalling time.

Bit 17  **SFLUSH**: S-FIFO flush (when I3C acts as controller)
This bit can be written and used only when I3C acts as controller.
0: no action
1: flush S-FIFO
Bit 16 **SDMAEN**: S-FIFO DMA request enable (when I3C acts as controller)

This bit must be cleared if SMODE = 0 in the I3C_CFRG register (S-FIFO is disabled). In other words, DMA mode cannot be used if S-FIFO is disabled. Then the status register I3C_SR can be read or not.

This bit can be set or cleared if SMODE = 1 (S-FIFO is enabled). In other words, status register I3C_SR must be read for each message, either by software, or via an allocated DMA channel.

0: DMA mode is disabled for reading status register I3C_SR
- SMODE = 0: software can read the I3C_SR register after a completed frame (FCF = 1 in the I3C_EV register) or an error (ERRF = 1 in the I3C_EV register). Via polling on these register flags or via interrupt notification (enabled by FCIE = 1 and ERRIE = 1 in the I3C_IER register).
- SMODE = 1: software must read and pop a status word from S-FIFO (read I3C_SR register) after each asserted flag SFNEF = 1. Via polling on this register flag or via interrupt notification (enabled by SFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for reading status register I3C_SR
- Must have SMODE = 1 in the I3C_CFRG register (S-FIFO enabled)
- DMA reads and pops status word(s) from S-FIFO (it reads I3C_SR register)
- Status word(s) are automatically read by the programmed hardware (via the asserted S-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TXTHRES**: TX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the TX-FIFO level, when the TXFNFF flag is set in the I3C_EV register (and consequently if TXDMAEN = 1 when is asserted a DMA TX request).

0: 1-byte threshold
- TXFNFF is set when 1 byte must be written in TX-FIFO (in I3C_TDR).

1: 1-word / 4-byte threshold
- TXFNFF is set when 1 word / 4 bytes must be written in TX-FIFO (in the I3C_TDWR register). If the a number of the last transmitted data is not a multiple of 4 bytes (XDCNT[1:0] = 00 in the I3C_SR register), only the relevant 1, 2, or 3 valid LSB bytes of the last word are taken into account by the hardware, and sent on the I3C bus.

Bit 13 **TXFLUSH**: TX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

When the I3C acts as target, this bit can be used to flush the TX-FIFO on a private read if the controller has aborted the data read (driven low the T bit), and there is/are remaining data in the TX-FIFO (ABT = 1, and XDCNT[15:0] in the I3C_SR register < TGTTDCNT[15:0] in the I3C_TGTTDR register).

0: no action
1: flush TX-FIFO

Bit 12 **TXDMAEN**: TX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for TX-FIFO
- Software writes and pushes a data byte/word into TX-FIFO (writes I3C_TDR or I3C_TDWR register), to be transmitted over the I3C bus.
- A next data byte/word must be written by the software either via polling on the flag TXFNFF = 1 or via interrupt notification (enabled by TXFNFIE = 1).

1: DMA mode is enabled for TX-FIFO
- DMA writes and pushes data byte(s)/word(s) into TX-FIFO (writes I3C_TDR or I3C_TDWR register).
- A next data byte/word transfer is automatically pushed by the programmed hardware (via the asserted TX-FIFO DMA request from the I3C and the programmed DMA channel).
Bit 11 Reserved, must be kept at reset value.

Bit 10 **RXTHRES**: RX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the RX-FIFO level, when the RXFNEF flag in the I3C_EVR register is set (and consequently if RXDMAEN = 1 when is asserted a DMA RX request).

0: 1-byte threshold
RXFNEF is set when 1 byte must be read in RX-FIFO (in the I3C_RDR register).

1: 1-word/4-bytes threshold
RXFNEF is set when 1 word / 4 bytes is/are to be read in RX-FIFO (in I3C_RDWR). In the case of a number of last received data being not a multiple of 4 bytes, only the relevant 1, 2 or 3 valid LSB bytes of the last word are to be considered by the software. The number of effective received data bytes is reported by XDCNT[15:0] in the I3C_SR register.

Bit 9 **RXFLUSH**: RX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

0: no action
1: flush RX-FIFO

Bit 8 **RXDMAEN**: RX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for RX-FIFO
- Software reads and pops a data byte/word from RX-FIFO (it reads I3C_RDR or I3C_RDWR register).
- A next data byte/word must be read by the software either via polling flag RXFNEF = 1 in the I3C_EVR register, or via interrupt notification (enabled by RXFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for RX-FIFO
- DMA reads and pops data byte(s)/word(s) from RX-FIFO (reads I3C_RDR or I3C_RDWR register).
- A next data byte/word is automatically read by the programmed hardware (via the asserted RX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 7 **HJACK**: Hot-join request acknowledge (when I3C acts as a controller)

0: hot-join request is not acknowledged
After the NACK, the controller continues as initially programmed (the hot-joining target is aware of the NACK and must emit another hot-join request later on).

1: hot-join request is acknowledged
After the ACK, the controller continues as initially programmed. The software is notified by the HJ interrupt (flag HJF is set in the I3C_EVR register), and must initiate the ENTDAA sequence later on, potentially preventing other hot-join requests with a disable target events command (DISEC, with DISHJ = 1).

Bit 6 Reserved, must be kept at reset value.

Bit 5 **HKSDAEN**: High-keeper enable on SDA line (when I3C acts as a controller)

0: High-keeper is disabled
1: High-keeper is enabled, and the weak pull-up is effective on the T bit, instead of the open-drain class pull-up.

*Note*: This bit can be modified only when EN = 0 in the I3C_CFG register.
Bit 4  **EXITPTRN**: HDR exit pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

- **0**: HDR exit pattern is not sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register). This is used to send the header, to test ownership of the bus when there is a suspicion of a problem after controller-role hand-off (new controller did not assert its controller-role by accessing the previous one in less than the delay defined by the activity state).
- **1**: HDR exit pattern is sent after the issued message header (MTYPE[3:0] = 0001). This is used on a controller error detection and escalation handling, in case of a not responding target to a private message or a direct read CCC.

The HDR exit pattern is sent whatever the message header (S/Sr + 0x7E addr + W) is ACK-ed or NACK-ed.

Bit 3  **RSTPTRN**: HDR reset pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

- **0**: standard stop emitted at the end of a frame
- **1**: HDR reset pattern is inserted before the stop of any emitted frame that includes a RSTACT CCC command

Bit 2  **NOARBH**: No arbitrable header after a start (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

- **0**: An arbitrable header (0b111_1110 + RnW = 0) is emitted after a start and before a legacy I2C message or an I3C SDR private read/write message (default).
- **1**: No arbitrable header
  - The target address is emitted directly after a start in case of a legacy I2C message or an I3C SDR private read/write message.
  - This is a more performing option (when the emission of the 0x7E arbitrable header is useless), but must be used only when the controller is sure that the addressed target device cannot emit concurrently an IBI or a controller-role request (to insure no misinterpretation and no potential conflict between the address emitted by the controller in open-drain mode and the same address a target device can emit after a start, for IBI or MR).

Bit 1  **CRINIT**: Initial controller/target role

This bit can be modified only when EN = 0 in the I3C_CFG register.

- **0**: target role
  - Once enabled by setting EN = 1, the peripheral initially acts as a target. I3C does not drive SCL line and does not enable SDA pull-up, until it eventually acquires the controller role.
- **1**: controller role
  - Once enabled by setting EN = 1, the peripheral initially acts as a controller. It has the I3C controller role, so drives SCL line and enables SDA pull-up, until it eventually offers the controller role to an I3C secondary controller.

Bit 0  **EN**: I3C enable (whatever I3C acts as controller/target)

- **0**: I3C is disabled
  - Except registers, the peripheral is under reset (partial reset).
  - Before clearing EN, when I3C acts as a controller, all the possible target requests must be disabled using DISEC CCC.
  - When I3C acts as a target, software must not disable the I3C, unless a partial reset is needed.
- **1**: I3C is enabled
  - In this state, some register fields cannot be modified (like CRINIT, HKSDAEN for the I3C_CFG)
49.16.4 I3C receive data byte register (I3C_RDR)

Address offset: 0x010
Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a byte-based read access (RXTHRES = 0 in the I3C_CFGR register). Then:

- On read, I3C_RDR returns a 32-bit data word, with the received data byte in LSB position, and other reserved bits read as 0.
- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must wait to be notified that a next received data byte must be read via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received byte of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO is full, a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by error flag ERRF = 1 in the I3C_EVR register and by a data overrun flag DOVRF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RDB[7:0]**: 8-bit received data on I3C bus.

49.16.5 I3C receive data word register (I3C_RDWR)

Address offset: 0x014
Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a 32-bit word-based read access (RXTHRES = 1 in the I3C_CFGR register). Then:

- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must first wait to be notified that must be read a next received data word via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received
word/byte(s) of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.

- If the RX-FIFO holds less than four bytes, a read on I3C_RDWR returns a data word padded with null byte(s): the available byte(s) in LSB position(s) is (are) padded with zero byte(s) in MSB position(s).
- If the RX-FIFO is full and a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data overrun DOVR = 1 in the I3C_SER register (and corresponding interrupt if enabled).

| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| RDB3[7:0] | RDB2[7:0] |
|      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |

Bits 31:24  **RDB3[7:0]**: 8-bit received data (latest byte on I3C bus).
Bits 23:16  **RDB2[7:0]**: 8-bit received data (next byte after RDB1 on I3C bus).
Bits 15:8  **RDB1[7:0]**: 8-bit received data (next byte after RDB0 on I3C bus).
Bits 7:0  **RDB0[7:0]**: 8-bit received data (earliest byte on I3C bus).

### 49.16.6 I3C transmit data byte register (I3C_TDR)

Address offset: 0x018

Reset value: 0x0000 0000

This register is used to write data bytes to be transmitted over the I3C bus.

This register implements a byte-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 0 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 0 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 0: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last transmitted byte of a given message to be written is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 0 and if PRELOAD = 1 in the I3C_TGTDDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to I3C_TDR register, according to TGTDDCNT[15:0] in the I3C_TGTDDR register.
- When TXDMAEN = 0 and if TXTHRES = 0:
  - when PRELOAD = 1 in the I3C_TGTDDR register: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non
full flag (TXFNFF = 1) or via the corresponding interrupt, if enabled. The last transmitted byte of the initially programmed TGTTDCNT[15:0] is also marked with TXLASTF = 1 in the I3C_EVR register.

- when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to 8 next data bytes (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. If the software needs to write another data byte, it must wait for the TX-FIFO to be empty (TXFEF = 1), and then write this data byte to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 (and corresponding interrupt if enabled) in the I3C_SER register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 TDB0[7:0]: 8-bit data to transmit on I3C bus.

### 49.16.7 I3C transmit data word register (I3C_TDWR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to write 32-bit data words to be transmitted over the I3C bus.

This register implements a word-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 1 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 1 in the I3C_CFGR register, programmed I3C and DMA automatically manage by hardware the relevant and successive writes.

- When TXDMAEN = 0 and if TXTHRES = 1: before a write, the software must wait to be notified that next data word/byte(s) must be written via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. Last transmitted word/byte(s) of a given message to be written in TX-FIFO is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 1 and if PRELOAD = 1 in the I3C_TGTTDDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to the I3C_TDWR register, as per TGTTDCNT[15:0]
in the I3C_TGTTDR register.

- When TXDMAEN = 0 and if TXTHRES = 1:
  - when PRELOAD = 1: before a write, the software must wait to be notified that must be written a next data word via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt, if enabled. Last transmitted word of the initially programmed TGTTDCNT[15:0] in the I3C_TGTTDR register is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to two next data words (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. If the software needs to write another data word, it must wait for TX-FIFO to be empty (TXFEF = 1), and then write the next data word to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller/target cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDB3[7:0] TDB2[7:0] TDB1[7:0] TDB0[7:0]</td>
</tr>
<tr>
<td>w  w  w  w  w  w  w  w  w  w  w  w  w  w  w</td>
</tr>
<tr>
<td>15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0</td>
</tr>
</tbody>
</table>

Bits 31:24 | **TDB3[7:0]**: 8-bit transmit data (latest byte on I3C bus).

Bits 23:16 | **TDB2[7:0]**: 8-bit transmit data (next byte after TDB1[7:0] on I3C bus).

Bits 15:8  | **TDB1[7:0]**: 8-bit transmit data (next byte after TDB0[7:0] on I3C bus).

Bits 7:0   | **TDB0[7:0]**: 8-bit transmit data (earliest byte on I3C bus)
**49.16.8 I3C IBI payload data register (I3C_IBIDR)**

Address offset: 0x020

Reset value: 0x0000 0000

This register is used for the IBI payload data.

When I3C acts as target:
- if BCR2 = 0 in the I3C_BCR register, this register is useless.
- if BCR2 = 1, this register must be written by software
  - to be emitted on the I3C bus as the IBI payload data, after that the IBI request (MTYPE[3:0] = 1010 in the I3C_CR register) is acknowledged by the controller; with the IBI data payload size defined by DCNT[15:0] in the I3C_CR register.
  - Maximum (static) payload data size is given by IBIP[2:0] in the I3C_MAXRLR register. it can be 1, 2, 3 or 4 bytes.
  - DCNT[15:0] must be set between 1 (for the mandatory data byte MDB[7:0]) and the maximum IBIP[2:0].

When I3C acts as controller: if IBIACK= 1 in the I3C_DEVRx register, once it acknowledges on the I3C bus the IBI request from the target x:
- if IBIDEN = 0 (BCR[2] = 0 received from target x) in the I3C_DEVRx register, this register is useless.
- if IBIDEN = 1 (BCR[2] = 1 received from target x):
  - This register is internally written by hardware from the IBI payload data received on the I3C bus (including the first mandatory data byte MDB[7:0]).
  - When the last byte of the payload is received in I3C_IBIDR (when target drives T-bit = 0), IBIF flag is set (and corresponding interrupt if enabled) in the I3C_EVR register.
  - Then, the software can identify the target x via the received and logged 7-bit address in RADD[6:0] in the I3C_RMR register.
  - The software can read this register and interpret byte(s) according to the number of bytes effectively received and logged in IBIRDCNT[2:0] in the I3C_RMR register.

<table>
<thead>
<tr>
<th>Address offset</th>
<th>Bit Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24</td>
<td>IBIDB3[7:0]: 8-bit IBI payload data (latest byte on I3C bus).</td>
</tr>
<tr>
<td>23 22 21 20 19 18 17 16</td>
<td>IBIDB2[7:0]: 8-bit IBI payload data (next byte on I3C bus after IBIDB1[7:0]).</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>IBIDB1[7:0]: 8-bit IBI payload data (next byte on I3C bus after IBIDB0[7:0]).</td>
</tr>
<tr>
<td>7:0</td>
<td>IBIDB0[7:0]: 8-bit IBI payload data (earliest byte on I3C bus, MDB[7:0] mandatory data byte).</td>
</tr>
</tbody>
</table>
49.16.9  I3C target transmit configuration register (I3C_TGTTDR)

Address offset: 0x024
Reset value: 0x0000 0000

When I3C acts as target, this register must be used to preload a number of data bytes in the TX-FIFO, so that they are ready to be transmitted on the I3C bus, when they are accepted/acknowledged by the controller, whatever the DMA mode is used or not (independently from TXDMAEN in the I3C_CFRG register).

When I3C acts as target, alternatively, if the number of data bytes to be transmitted is less than or equal to the TX-FIFO size (8 bytes), the software can directly use and write byte(s) into the I3C_TDR register (or I3C_TDWR register, depending upon TXTHRES in the I3C_CFRG register), via polling on the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding enabled interrupt.

In any case, since an I3C target is unable to stretch/stall the SCL line, the software can identify a TX-FIFO underrun via DOR = 1 in the I3C_EVR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TGTTCNT[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:17  Reserved, must be kept at reset value.

Bit 16  PRELOAD: Preload of the TX-FIFO (when I3C is configured as target)
This bit must be written and asserted by software in the same access when is written and defined the number of bytes to preload into the TX-FIFO and to transmit.
This bit is cleared by hardware when all the data bytes to transmit are loaded into the TX-FIFO.
0: no TX-FIFO preload
1: TX-FIFO preload

Bits 15:0  TGTTCNT[15:0]: Transmit data counter, in bytes (when I3C is configured as target)
This bitfield must be written by software in the same access when is asserted PRELOAD, in order to define the number of bytes to preload and to transmit.
This bitfield is updated by hardware and reports, when read, the remaining number of bytes to be loaded into the TX-FIFO.
49.16.10 I3C status register (I3C_SR)

Address offset: 0x030
Reset value: 0x0000 0000

This register is used to read the status about the exchanged message on the I3C bus:

- in FIFO mode: when the I3C acts as controller and if S-FIFO is enabled via SMODE = 1 in the I3C_CFGR register:
  - Software is notified via SFNEF = 1 in the I3C_EVR register if there is a status register to be read (and corresponding interrupt if enabled), when not in DMA mode (SDMAEN = 0)
  - In DMA mode (SDMAEN = 1), programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
  - Software is notified via COVR = 1 in the I3C_SER register on an S-FIFO overflow (and ERRF = 1 in the I3C_EVR register and corresponding interrupt if enabled)

- in register mode: if SMODE = 0 in the I3C_CFGR register
  - Software can use the flags FCF in the I3C_SER register and ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled) to read this register
  - This register can be overwritten by hardware on a new message completion, without any notification.

### Bits 31:24 MID[7:0]
Message identifier/counter of a given frame (when the I3C acts as controller)
When I3C acts as controller, this bitfield identifies the control word message (I3C_CR) to whom the I3C_SR status register refers.
First message of a frame is identified with MID[7:0] = 0.
This bitfield is incremented (by hardware) on the completion of a new message control word (I3C_CR) over I3C bus. This field is reset for every new frame start.

### Bits 23:19 Reserved, must be kept at reset value.

### Bit 18 DIR: Message direction
Whatever the I3C acts as controller or target, this bit indicates the direction of the related message on the I3C bus
0: write
1: read

**Note:** ENTDAA CCC is considered as a write command.
Bit 17 **ABT**: A private read message is ended prematurely by the target (when the I3C acts as controller)

When the I3C acts as controller, this bit indicates if the private read data transmitted by the target early terminates (the target drives T bit low earlier vs. what the controller expects in terms of programmed number of read data bytes DCNT[15:0] in the I3C_CR register).

0: no early completion from the target
1: early completion from the target

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **XDCNT[15:0]**: Data counter

Condition: during the dynamic address assignment process (ENTDAA CCC)
- When the I3C acts as controller: number of targets detected on the bus
- When the I3C acts as target: number of transmitted bytes

Condition: for other transfers, during the message
- Whatever the I3C acts as controller or target: number of data bytes read from or transmitted on the I3C bus during the message

### 49.16.11 I3C status error register (I3C_SER)

Address offset: 0x034

Reset value: 0x0000 0000

This read register is used to get more information about the error when an error is raised by hardware and notified to the software via the error flag ERF = 1 in the I3C_EVR register (and corresponding interrupt if enabled).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DERR</th>
<th>DNACK</th>
<th>ANACK</th>
<th>COVR</th>
<th>DOVR</th>
<th>STALL</th>
<th>PERR</th>
<th>CODERR[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DERR**: Data error (when the I3C acts as controller)

0: no detected error
1: controller detected a data error during the controller-role hand-off procedure (GETACCCR CCC, formerly known as GETACCMST) when the received target address or/and the parity bit do no match. Active controller keeps controller-role.

Bit 9 **DNACK**: Data not acknowledged (when the I3C acts as controller)

0: no detected error
1: controller detected that a data byte is not acknowledged by a target, either during:
   i) a legacy I2C write transfer
   ii) the second trial when sending dynamic address during ENTDAA procedure
Bit 8 **ANACK**: Address not acknowledged (when the I3C is configured as controller)
   0: no detected error
   1: controller detected that the static/dynamic address was not acknowledged by a target, either during:
      i) a legacy I2C read/write transfer
      ii) a direct CCC write transfer
      iii) the second trial of a direct CCC read transfer
      iv) a private read/write transfer

Bit 7 **COVR**: C-FIFO underrun or S-FIFO overrun (when the I3C acts as controller)
   0: no detected error
   1: controller detected either:
      i) a C-FIFO underrun: control FIFO is empty and a restart must be emitted
      ii) an S-FIFO overrun: S-FIFO is full and a new message ends

Bit 6 **DOVR**: RX-FIFO overrun or TX-FIFO underrun
   0: no detected error
   1: whatever controller or target, hardware detected either:
      i) a TX-FIFO underrun: TX-FIFO is empty and a write data byte must be transmitted
      ii) a RX-FIFO overrun: RX-FIFO is full and a new data byte is received

Bit 5 **STALL**: SCL stall error (when the I3C acts as target)
   0: no detected error
   1: target detected that SCL was stable for more than 125 µs during an I3C SDR data read
      (during a direct CCC read, a private read, or an IB)

Bit 4 **PERR**: Protocol error
   0: no detected error
   1: whatever controller or target, hardware detected a protocol error, as detailed in CODERR[3:0]
Bits 3:0 **CODERR[3:0]**: Protocol error code/type
- 0000: CE0 error (transaction after sending CCC): controller detected an illegally formatted CCC
- 0001: CE1 error (monitoring error):
  - controller detected that transmitted data on the bus is different from expected
- 0010: CE2 error (no response to broadcast address):
  - controller detected a not acknowledged broadcast address (0b111_1110)
- 0011: CE3 error (failed controller-role hand-off):
  - controller detected the new controller did not drive bus after controller-role hand-off
- 1000: TE0 error (invalid broadcast address 0b111_1110 + W):
  - target detected an invalid broadcast address 0b111_1110 + W
- 1001: TE1 error (CCC code):
  - target detected a parity error on a CCC code via a parity check (vs. T bit)
- 1010: TE2 error (write data):
  - target detected a parity error on a write data via a parity check (vs. T bit)
- 1011: TE3 error (assigned address during dynamic address arbitration):
  - target detected a parity error on the assigned address during dynamic address arbitration via a parity check (vs. PAR bit)
- 1100: TE4 error (0b111_1110 + R missing after Sr during dynamic address arbitration):
  - target detected a 0b111_1110 + R missing after Sr during dynamic address arbitration
- 1101: TE5 error (transaction after detecting CCC):
  - target detected an illegally formatted CCC
- 1110: TE6 error (monitoring error):
  - target detected that transmitted data on the bus is different from expected
- others: reserved

### 49.16.12 I3C received message register (I3C_RMR)

Address offset: 0x040

Reset value: 0x0000 0000

When the I3C acts as controller, this read register is used to log the received target address, and the IBI received payload data size.

When the I3C acts as target, this read register is used to log the received CCC code

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **RADD[6:0]**: Received target address (when the I3C is configured as controller)
  - When the I3C is configured as controller, this field logs the received dynamic address from the target during acknowledged IBI or controller-role request.
  - Bit 16: Reserved, must be kept at reset value.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

- **RCODE[7:0]**: Received CCC code
- **IBIRDcnt[2:0]**: Received parity bit count

- Bits 31:24: Reserved, must be kept at reset value.

- Bits 23:17: **RADD[6:0]**: Received target address (when the I3C is configured as controller)
  - When the I3C is configured as controller, this field logs the received dynamic address from the target during acknowledged IBI or controller-role request.

- Bit 16: Reserved, must be kept at reset value.
**Improved inter-integrated circuit (I3C)**

**49.16.13 I3C event register (I3C_EVR)**

Address offset: 0x050
Reset value: 0x0000 0003

This is a read register, used for reporting event flags.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRPF</td>
<td>DEFF</td>
<td>INT_UPDF</td>
<td>AS_UPDF</td>
<td>RSTF</td>
<td>MRL_UPDF</td>
<td>WKL_UPDF</td>
<td>DA_UPDF</td>
<td>STAF</td>
<td>GETF</td>
<td>WKPFD</td>
<td>HJF</td>
<td>CR_UPDF</td>
<td>CRF</td>
<td>IBI_ENDF</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **GRPF**: Group addressing flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller-capable), this flag is asserted by hardware to indicate that the broadcast DEFGRPA CCC (define list of group addresses) has been received. Then, software can store the received data for when getting controller role.
The flag is cleared when software writes 1 into the corresponding CGRPF bit in the I3C_CR register.

Bit 30 **DEFF**: DEFTGTS flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller capable), this flag is asserted by hardware to indicate that the broadcast DEFTGTS CCC (define list of targets) has been received. Then, software can store the received data for when getting the controller role.
The flag is cleared when software writes 1 into the corresponding CDEFF bit in the I3C_CEVR register.

Bit 29 **INTUPDF**: Interrupt/controller-role/hot-join update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENEC/DISEC CCC (enable/disable target events) has been received, where a target event is either an interrupt/IBI request, a controller-role request, or an hot-join request. Then, software must read respectively IBIEN, CREN, or HJEN in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CINTUPDF bit in the I3C_CEVR register.

Bit 28 **ASUPDF**: Activity state update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENASx CCC (with x = 0...3) has been received. Then, software must read AS[1:0] in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CASUPDF bit in the I3C_CEVR register.
Bit 27 **RSTF:** Reset pattern flag (when the I3C acts as target)

When I3C acts as target, this flag is asserted by hardware to indicate that a reset pattern has been detected (14 SDA transitions while SCL is low, followed by repeated start, then stop). Then, when not in Stop mode, software must read RSTACT[1:0] and RSTVAL in the I3C_DEVR0 register, to know the required reset level.

- If RSTVAL = 1: when the RSTF is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
- If RSTVAL = 0: when the RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.

The flag is cleared when software writes 1 into the corresponding CRSTF bit in the I3C_CEVR register.

Bit 26 **MRLUPDF:** Maximum read length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMRL CCC (set max read length) has been received. Then, software must read MRL[15:0] in the I3C_MAXRLR register to get the maximum read length value.

The flag is cleared when software writes 1 into the corresponding CMRLUPDF bit in the I3C_CEVR register.

Bit 25 **MWLUPDF:** Maximum write length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMWL CCC (set max write length) has been received. Then, software must read MWL[15:0] in the I3C_MAXRLR register to get the maximum write length value.

The flag is cleared when software writes 1 into the corresponding CMWLUPDF bit in the I3C_CEVR register.

Bit 24 **DAUPDF:** Dynamic address update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a dynamic address update has been received via any of the broadcast ENTDAA, RSTDAA and direct SETNEWDA CCC. Then, software must read DA[6:0] and DAVAL in the I3C_DEVR0 register to get the dynamic address update.

The flag is cleared when software writes 1 into the corresponding CDAUPDF bit in the I3C_CEVR register.

Bit 23 **STAF:** Get status flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct GETSTATUS CCC of format 1 (without defining byte or with defining byte TGTSTAT) has been received.

The flag is cleared when software writes 1 into the corresponding CSTAF bit in the I3C_CEVR register.

Bit 22 **GETF:** Get flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that any direct CCC of get type (GET*** CCC) except the GETSTATUS of format 1 (but including GETSTATUS of format 2) has been received.

The flag is cleared when software writes 1 into the corresponding CGETF bit in the I3C_CEVR register.
Bit 21 **WKPF**: Wake-up/missed start flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that a start has been detected (an SDA falling edge followed by an SCL falling edge) but on the next SCL falling edge, the I3C kernel clock is (still) gated. Thus an I3C bus transaction may have been lost by the target.
The corresponding interrupt can be used to wake up the device from a low power (Sleep or Stop) mode.
The flag is cleared when software writes 1 into the corresponding CWKPF bit in the I3C_CEVR register.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJF**: Hot-join flag (when the I3C acts as controller)
When the I3C acts as controller, this flag is asserted by hardware to indicate that an hot join request has been received.
The flag is cleared when software writes 1 into the corresponding CHJF bit in the I3C_CEVR register.

Bit 18 **CRUPDF**: Controller-role update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that it has now gained the controller role after the completed controller-role hand-off procedure.
The flag is cleared when software writes 1 into the corresponding CCRUPDF bit in the I3C_CEVR register.

Bit 17 **CRF**: Controller-role request flag (when the I3C acts as controller)
When the I3C acts as controller, this flag is asserted by hardware to indicate that a controller-role request has been acknowledged and completed (by hardware). The software must then issue a GETACCCR CCC (get accept controller role) for the controller-role hand-off procedure.
The flag is cleared when software writes 1 into the corresponding CCRF bit in the I3C_CEVR register.

Bit 16 **IBIENDF**: IBI end flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that an IBI transfer has been received and completed (IBI acknowledged and IBI data bytes read by controller if any).
The flag is cleared when software writes 1 into the corresponding CIBIENDF bit in the I3C_CEVR register.

Bit 15 **IBIF**: IBI flag (when the I3C acts as controller)
When the I3C acts as controller, this flag is asserted by hardware to indicate that an IBI request has been received.
The flag is cleared when software writes 1 into the corresponding CIBIF bit in the I3C_CEVR register.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRF**: Flag (whatever the I3C acts as controller/target)
This flag is asserted by hardware to indicate that an error occurred. Then, software must read I3C_SER to get the error type.
The flag is cleared when software writes 1 into the corresponding CERRF bit in the I3C_CEVR register.
Bit 10 **RXTGTENDF**: Target-initiated read end flag (when the I3C acts as controller)
When the I3C acts as controller, and only if the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register), this flag is asserted by hardware to indicate that the target has prematurely ended a read transfer. Then, software must read the status register I3C_SR to check information related to the last message and get the number of received data bytes on the prematurely read transfer (XDCNT in the I3C_SR register).
The flag is cleared when software writes 1 into the corresponding CRXTGTENDF bit in the I3C_CEVR register.

Bit 9 **FCF**: Frame complete flag (whatever the I3C acts as controller/target)
When the I3C acts as controller, this flag is asserted by hardware to indicate that a frame has been (normally) completed on the I3C bus, for example, when a stop is issued.
When the I3C acts as target, this flag is asserted by hardware to indicate that a message addressed to/by this target has been (normally) completed on the I3C bus, for example, when a next stop or repeated start is then issued by the controller.
The flag is cleared when software writes 1 into the corresponding CFCF bit in the I3C_CEVR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **RXLASTF**: Last read data byte/word flag (when the I3C acts as controller)
When the I3C acts as controller, this flag is asserted by hardware to indicate that the last data byte/word (depending upon RXTHRES in the I3C_CFGR register) of a message must be read from the RX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is read.

Bit 6 **TXLASTF**: Last written data byte/word flag (whatever the I3C acts as controller/target)
This flag is asserted by hardware to indicate that the last data byte/word (depending upon TXTHRES in the I3C_CFGR register) of a message must be written to the TX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is written.

Bit 5 **RXFNEF**: RX-FIFO not empty flag (whatever the I3C acts as controller/target)
This flag is asserted/de-asserted by hardware to indicate that a data byte must/must not be read from the RX-FIFO.
*Note*: The software must wait for RXFNEF = 1 (by polling or via the enabled interrupt) before reading from RX-FIFO (reading from I3C_RDR or I3C_RDWR, depending upon RXTHRES).

Bit 4 **TXFNFF**: TX-FIFO not full flag (whatever the I3C acts as controller/target)
This flag is asserted/de-asserted by hardware to indicate that a data byte word must/must not be written to the TX-FIFO.
*Note*: The software must wait for TXFNFF = 1 (by polling or via the enabled interrupt) before writing to TX-FIFO (writing to I3C_TDR or I3C_TDWR, depending upon TXTHRES).
*Note*: When the I3C acts as target, if the software intends to use the TXFNFF flag for writing into I3C_TDR/I3C_TDWR, it must have configured and set the TX-FIFO preload (write PRELOAD in the I3C_TGTTDR register).

Bit 3 **SFNEF**: S-FIFO not empty flag (when the I3C acts as controller)
When the I3C acts as controller, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), this flag is asserted by hardware to indicate that a status word must be read from the S-FIFO. The flag is de-asserted by hardware to indicate that a status word is not to be read from the S-FIFO.
Improved inter-integrated circuit (I3C) RM0481

49.16.14 I3C interrupt enable register (I3C_IER)

Address offset: 0x054

Reset value: 0x0000 0000

This register is used to enable/disable, at bit level, an interrupt, for each of the following event/flag.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRPIE</td>
<td>DEFIE</td>
<td>INTUPDIE</td>
<td>ASLUPDIE</td>
<td>RSTIE</td>
<td>MRLUPDIE</td>
<td>MMXUPDIE</td>
<td>NAMPE</td>
<td>STAIE</td>
<td>GETIE</td>
<td>TPMPE</td>
<td>PABIE</td>
<td>FJIE</td>
<td>CRUPDIE</td>
<td>CRPE</td>
<td>BIENIE</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bit 31 **GRPIE**: DEFGRPA CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 30 **DEFIE**: DEFTGTS CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 29 **INTUPDIE**: ENEC/DISEC CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled
<table>
<thead>
<tr>
<th>Bit 28</th>
<th>ASUPDIE: ENTASx CCC interrupt enable (when the I3C acts as target)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 27</td>
<td>RSTIE: reset pattern interrupt enable (when the I3C acts as target)</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 26</td>
<td>MRLUPDIE: SETMRL CCC interrupt enable (when the I3C acts as target)</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 25</td>
<td>MWLUPDIE: SETMWL CCC interrupt enable (when the I3C acts as target)</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 24</td>
<td>DAUPDIE: ENTDAA/RSTDAA/SETNEWDA CCC interrupt enable (when the I3C acts as target)</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 23</td>
<td>STAIE: format 1 GETSTATUS CCC interrupt enable (when the I3C acts as target)</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 22</td>
<td>GETIE: GETxxx CCC interrupt enable (except GETSTATUS of format 1) (when the I3C acts as target)</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 21</td>
<td>WKPIE: Wake-up interrupt enable (when the I3C acts as target)</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 19</td>
<td>HJIE: Hot-join interrupt enable (when the I3C acts as controller)</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 18</td>
<td>CRUPDIE: Controller-role update interrupt enable (when the I3C acts as target)</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 17</td>
<td>CRIE: Controller-role request interrupt enable (when the I3C acts as controller)</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 16</td>
<td>IBIENDIE: IBI end interrupt enable (when the I3C acts as target)</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bit 15</td>
<td>IBIIE: IBI request interrupt enable (when the I3C acts as controller)</td>
</tr>
<tr>
<td></td>
<td>0: interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: interrupt enabled</td>
</tr>
<tr>
<td>Bits 14:12</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bit 11 **ERRIE**: error interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 10 **RXTGTENIE**: target-initiated read end interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 9 **FCIE**: frame complete interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEIE**: RX-FIFO not empty interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 4 **TXFNFIE**: TX-FIFO not full interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 3 **SFNEIE**: S-FIFO not empty interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bit 2 **CFNFIE**: C-FIFO not full interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bits 1:0 Reserved, must be kept at reset value.

### 49.16.15 I3C clear event register (I3C_CEVR)

Address offset: 0x058
Reset value: 0x0000 0000

This write register is used to clear individually, at bit level, the corresponding event flag of the I3C_CEVR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGPF</td>
<td>CBIF</td>
<td>CDEFF</td>
<td>CINTLPDF</td>
<td>CASLPDF</td>
<td>CRSTF</td>
<td>CMRLUPDF</td>
<td>CMMLUPDF</td>
<td>CCDF</td>
<td>CGETF</td>
<td>CVKF</td>
<td>CHUF</td>
<td>CCRUPDF</td>
<td>CCRF</td>
<td>CBENDF</td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**CBIF**: Pass

**CDEFF**: Pass

**CINTLPDF**: Pass

**CASLPDF**: Pass

**CRSTF**: Pass

**CMRLUPDF**: Pass

**CMMLUPDF**: Pass

**CCDF**: Pass

**CGETF**: Pass

**CVKF**: Pass

**CHUF**: Pass

**CCRUPDF**: Pass

**CCRF**: Pass

**CBENDF**: Pass

Bit 31 **CGRF**: Clear DEFGRPA CCC flag (when the I3C acts as target)
0: no effect
1: clear GRPF
Bit 30 **CDEFF**: Clear DEFTGTS CCC flag (when the I3C acts as target)
0: no effect
1: clear DEFF

Bit 29 **CINTUPDF**: Clear ENEC/DISEC CCC flag (when the I3C acts as target)
0: no effect
1: clear CINTUPDF

Bit 28 **CASUPDF**: Clear ENTASx CCC flag (when the I3C acts as target)
0: no effect
1: clear ASUPDF

Bit 27 **CRSTF**: Clear reset pattern flag (when the I3C acts as target)
0: no effect
1: clear RSTF

Bit 26 **CMRLUPDF**: Clear SETMRL CCC flag (when the I3C acts as target)
0: no effect
1: clear MRLUPDF

Bit 25 **CMWLUPDF**: Clear SETMWL CCC flag (when the I3C acts as target)
0: no effect
1: clear MWLUPDF

Bit 24 **CDAUPDF**: Clear ENTDAA/RSTDAA/SETNEWDA CCC flag (when the I3C acts as target)
0: no effect
1: clear DAUPDF

Bit 23 **CSTAF**: Clear format 1 GETSTATUS CCC flag (when the I3C acts as target)
0: no effect
1: clear STAF

Bit 22 **CGETF**: Clear GETxxx CCC flag (except GETSTATUS of format 1) (when the I3C acts as target)
0: no effect
1: clear GETF

Bit 21 **CWKPF**: Clear wake-up flag (when the I3C acts as target)
0: no effect
1: clear WKPF

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CHJF**: Clear hot-join flag (when the I3C acts as controller)
0: no effect
1: clear HJF

Bit 18 **CCRUPDF**: Clear controller-role update flag (when the I3C acts as target)
0: no effect
1: clear CRUPDF

Bit 17 **CCRF**: Clear controller-role request flag (when the I3C acts as controller)
0: no effect
1: clear CRF

Bit 16 **CIBIENDF**: Clear IBI end flag (when the I3C acts as target)
0: no effect
1: clear IBIENDF
Bit 15 **CIBIF**: Clear IBI request flag (when the I3C acts as controller)
0: no effect
1: clear IIBIF

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **CERRF**: Clear error flag (whatever the I3C acts as controller/target)
0: no effect
1: clear ERRF

Bit 10 **CRXTGTENDF**: Clear target-initiated read end flag (when the I3C acts as controller)
0: no effect
1: clear RXTGTENDF

Bit 9 **CFCF**: Clear frame complete flag (whatever the I3C acts as controller/target)
0: no effect
1: clear FCF

Bits 8:0 Reserved, must be kept at reset value.

### 49.16.16 I3C own device characteristics register (I3C_DEVR0)

Address offset: 0x060

Reset value: 0x0000 0000

When the I3C peripheral acts as target, this register is used to write or read its own device characteristics.

When the I3C peripheral acts as controller, the field DA[6:0] is used to write and store its own dynamic address.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Reset Value</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>RSTVAL: Reset action is valid (when the I3C acts as target)</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>RSTACT[1:0]: Reset action (target)</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>HJEN: High jump enable</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>AS[1:0]: Address select</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>reserved</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RSTVAL**: Reset action is valid (when the I3C acts as target)

This bit is asserted by hardware to indicate that the RSTACT[1:0] field has been updated on the reception of a broadcast or direct write RSTACT CCC (target reset action) and is valid. This bit is cleared by hardware when the target receives a frame start.

When the device is not in Stop mode:
- If RSTVAL = 1: when RSTF in the I3C_EVTR register is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
- If RSTVAL = 0: when RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.
Bits 23:22 RSTACT[1:0]: Reset action/level on received reset pattern (when the I3C acts as target)

This read field is used by hardware on the reception of a direct read RSTACT CCC in order to return the corresponding data byte on the I3C bus.

This read field is updated by hardware on the reception of a broadcast or direct write RSTACT CCC (target reset action).

Only the defining bytes 0x00, 0x01 and 0x02 are mapped, and RSTACT[1:0] = Defining Byte[1:0].

00: no reset action

01: first level of reset: the application software must either:

a) partially reset the peripheral, by a write and clear of the enable bit of the I3C configuration register (write EN = 0). This resets the I3C bus interface and the I3C kernel sub-parts, without modifying the content of the I3C APB registers (except the EN bit).

b) fully reset the peripheral, including all its registers, via a write and set of the I3C reset control bit of the RCC (reset and clock controller) register.

10: second level of reset: the application software must issue a warm reset, also known as a system reset. This (see Section 11: Reset and clock control (RCC)) has the same impact as a pin reset (NRST = 0):

– the software writes and sets the SYSRESETREQ control bit of the AITR register, when the device is controlled by a Cortex®-M.

– the software writes and sets SYSRST = 1 in the RCC_GRSTCSETR register, when the device is controlled by a Cortex®-A.

11: no reset action

Bits 21:20 AS[1:0]: Activity state (when the I3C acts as target)

This read field is updated by hardware on the reception of a ENTASx CCC (enter activity state, with x = 0-3):

00: activity state 0

01: activity state 1

10: activity state 2

11: activity state 3

Bit 19 HJEN: Hot-join request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISHJ= 1 (cleared) and the reception of ENEC CCC with ENHJ= 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: hot-join request disabled

1: hot-join request enabled

Bit 18 Reserved, must be kept at reset value.

Bit 17 CREN: Controller-role request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISCR = 1 (cleared) and the reception of ENEC CCC with ENCR = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: controller-role request disabled

1: controller-role request enabled
Bit 16 IBIEN: IBI request enable (when the I3C acts as target)
This bit is initially written by software when EN = 0, and is updated by hardware on the
reception of DISEC CCC with DISINT = 1 (cleared) and the reception of ENEC CCC with
ENINT = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR
register.
0: IBI request disabled
1: IBI request enabled

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 DA[6:0]: 7-bit dynamic address
When the I3C acts as controller, this field can be written by software, for defining its own
dynamic address.
When the I3C acts as target, this field is updated by hardware on the reception of either the
broadcast ENTDAA CCC or the direct SETNEWDA CCC.

Bit 0 DAVAL: Dynamic address is valid (when the I3C acts as target)
When the I3C acts as controller, this bit can be written by software, for validating its own
dynamic address, for example before a controller-role hand-off.
When the I3C acts as target, this bit is asserted by hardware on the acknowledge of the
broadcast ENTDAA CCC or the direct SETNEWDA CCC, and this field is cleared by
hardware on the acknowledge of the broadcast RSTDAA CCC.

49.16.17 I3C device x characteristics register (I3C_DEVRx)
Address offset: 0x060 + 0x4 * x, (x = 1 to 4)
Reset value: 0x0000 0000
When the I3C peripheral acts as controller, this register is used to define and store some
characteristics of a device target x with their related management from the controller, to
communicate accordingly with any of this target x over the I3C bus. Then, the hardware can
autonomously identify and acknowledge an allowed IBI or/and controller-role request from a
target x, receive the expected IBI payload data if any, and notify the software via the
corresponding flag IBIF/CRF (and the corresponding interrupt if enabled) in the I3C_EVR
register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31 DIS: DA[6:0] write disabled (when the I3C acts as controller)
When the I3C acts as controller, once the software sets IBIACK= 1 or CRACK= 1, this read
bit is set by hardware (DIS = 1) to lock the configured DA[6:0] and IBIDEN values.
Then, to be able to modify DA[6:0], IBIDEN or SUSP, the software must wait for DIS to be
de-asserted by hardware (polling on DIS = 0) before modifying these three assigned values
to the target x. Indeed, the target can request an IBI or a controller-role while the controller
intends to modify DA[6:0], IBIDEN or SUSP.
0: write to DA[7:0] and to IBIDEN in the I3C_DEVRx register is allowed
1: write to DA[7:0] and to IBIDEN is disabled/locked
Bits 30:20  Reserved, must be kept at reset value.

Bit 19  **SUSP**: Suspend/stop I3C transfer on received IBI (when the I3C acts as controller)

When the I3C acts as controller, this bit can be used to receive an IBI from target x with pending read notification feature (received MDB[7:5] = 3'b101).

If this bit is set, when an IBI is received and completed (IBIF = 1 in the I3C_EVR register), a stop is emitted on the I3C bus and both C-FIFO and TX-FIFO are automatically flushed by hardware. The controller execution flow is stopped, even if a next control message is programmed. When the IBI is completed, the controller software can issue a new control word, such as a private read, to the target device that initiated the IBI request.

0: C-FIFO and TX-FIFO are not flushed after an IBI request from target x is acknowledged and completed, and depending on the presence or absence of a next control word, a repeated start or a stop is emitted.
1: I3C transfer is stopped and both C-FIFO and TX-FIFO are flushed after receiving an IBI request from target x.

Bit 18  **IBIDEN**: IBI data enable (when the I3C acts as controller)

When the I3C acts as controller, this bit must be written by software to store the BCR[2] bit as received from the target x during broadcast ENTDAA or direct GETBCR CCC via the received I3C_RDR.

Writing to this field has no impact when the DIS = 1 in the I3C_DEVRx register.

0: no data byte follows the acknowledged IBI from target x.
1: the mandatory data byte MDB[7:0] follows the acknowledged IBI from target x.

Bit 17  **CRACK**: Controller-role request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of a controller-role request from target x:

0: a controller-role request from target x must be NACK-ed

After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another controller-role request later on).

1: a controller-role request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.

- After the ACK, the message continues as initially programmed. The software is notified by the controller-role request flag (CRF = 1 in the I3C_EVR register) and/or the corresponding interrupt if enabled; For effectively granting the controller-role to the requesting secondary controller, software must issue a GETACCCR (formerly known as GETACCMST), followed by a stop.

- Independently of CRACK configuration for this or other devices, further controller-role request(s) are NACK-ed until controller-role request flag (CRF) and IBI flag (IBIF) in the I3C_EVR register are both cleared.
Device: Improved inter-integrated circuit (I3C) RM0481

**49.16.18 I3C maximum read length register (I3C_MAXRLR)**

Address offset: 0x090

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum read length value exchanged with the controller during respectively GETMRL or SETMRL CCC. This register is also used to set the IBI data payload size.

This register can be written by the software when EN = 0 in the I3C_CFR register.

When receiving a private read message, the target ends the data transmission (by driving T-bit = 0) when the count of transmitted data reaches MRL[15:0] in the I3C_MAXRLR register.

| Bit 31:19 | Reserved, must be kept at reset value. |
| Bit 18:17 | IBIP[2:0] |
| Bits 16 | IBIACK: IBI request acknowledge (when the I3C acts as controller) |
| When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of an IBI request from target x: |
| 0: an IBI request from target x must be NACK-ed |
| - After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another IBI request later on) |
| 1: an IBI request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed |
| - The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain. |
| - After the ACK, the controller logs the IBI payload data, if any, depending on I3C DEVRx.IBIDEN. |
| - The software is notified by the IBI flag (IBIF = 1) and/or the corresponding interrupt if enabled; |
| - Independently from IBIACK configuration for this or other devices, further IBI request(s) are NACK-ed until IBI request flag (IBIF) and controller-role request flag (CRF) are both cleared. |
| Bits 15:8 | Reserved, must be kept at reset value. |
| Bits 7:1 | DA[6:0]: Assigned I3C dynamic address to target x (when the I3C acts as controller) |
| When the I3C acts as controller, this field must be written by software to store the 7-bit dynamic address that the controller sends via a broadcast ENTDAA or a direct SETNEWDA CCC acknowledged by the target x. |
| Writing to this field has no impact when the read field DIS = 1 in the I3C DEVRx register. |
| Bit 0 | Reserved, must be kept at reset value. |
**Improved inter-integrated circuit (I3C)**

49.16.19 **I3C maximum write length register (I3C_MAXWLR)**

Address offset: 0x094  
Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum write length value exchanged with the controller during respectively GETMWL or SETMWL CCC.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

On receiving a private write message, the target stops the data reception (extra received data are not written into RX-FIFO) when the count of received data reaches MWL[15:0] in the I3C_MAXWLR register.

| Bits 31:16 | Reserved | must be kept at reset value. |
| Bits 15:0  | MWL[15:0] | Maximum data write length (when I3C acts as target) |

Other details about the register and its usage are provided in the document.
49.16.20 I3C timing register 0 (I3C_TIMINGR0)

Address offset: 0x0A0

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure the SCL clock signal waveform.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

<table>
<thead>
<tr>
<th>Bits 31:24</th>
<th>SCLH_I2C[7:0]</th>
<th>SCLL_OD[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
</tr>
</tbody>
</table>

Bits 31:24 **SCLH_I2C[7:0]**: SCL high duration, used for legacy I2C messages, in number of kernel clocks cycles:

\[ \text{ts}_{\text{SCLH I2C}} = (\text{SCLH I2C} + 1) \times \text{tI3CCLK} \]

Note: SCLH I2C is used to generate \( t_{\text{DIG H (I2C)}} \) timing when communicating with I2C devices.

Note: With I2C fm+ device \( t_{\text{DIG H min}} = 260 \text{ ns} \), with I2C fm device \( t_{\text{DIG H min}} = 600 \text{ ns} \).

Bits 23:16 **SCLL_OD[7:0]**: SCL low duration in open-drain phases, used for legacy I2C messages and for I3C open-drain phases (address phase following a start, ACK phase during controller-initiated messages, and T bit phase during direct/private/IBI payload), in number of kernel clocks cycles:

\[ \text{ts}_{\text{SCLL OD}} = (\text{SCLL OD} + 1) \times \text{tI3CCLK} \]

Note: SCLL OD is used to generate both \( t_{\text{DIG L (I2C)}} \) and \( t_{\text{DIG OD L (I3C)}} \) timings.

Note: With I2C fm+ device \( t_{\text{DIG L min}} = 500 \text{ ns} \), with I2C fm device \( t_{\text{DIG L min}} = 1320 \text{ ns} \).

Note: I3C messages: \( t_{\text{DIG OD L min}} = 200 \text{ ns} \).

Note: If a single I3C frame is gathering I2C and I3C messages, the SCL low duration during I3C open-drain phases is increased to fit I2C timings.

Bits 15:8 **SCLH_I3C[7:0]**: SCL high duration, used for I3C messages (both in push-pull and open-drain phases), in number of kernel clocks cycles:

\[ \text{ts}_{\text{SCLH I3C}} = (\text{SCLH I3C} + 1) \times \text{tI3CCLK} \]

Note: SCLH I3C is used to generate both \( t_{\text{DIG H (I3C)}} \) and \( t_{\text{DIG H MIXED}} \) timings.

Note: For mixed bus (with at least one I2C target): \( t_{\text{DIG H MIXED min}} = 32 \text{ ns} \) and \( t_{\text{DIG H MIXED max}} = 45 \text{ ns} \) (due to I2C 50 ns spike filter).

Note: For pure I3C bus (with no I2C targets): \( t_{\text{DIG H min}} = 32 \text{ ns} \).

Bits 7:0 **SCLL_PP[7:0]**: SCL low duration in I3C push-pull phases, in number of kernel clocks cycles:

\[ \text{ts}_{\text{SCLL PP}} = (\text{SCLL PP} + 1) \times \text{tI3CCLK} \]

Note: SCCLL PP is used to generate \( t_{\text{DIG L (I3C in PP)}} \) timing.

Note: \( t_{\text{DIG L min}} = 32 \text{ ns} \) (max 40/60 duty cycle at 12.5 MHz).
49.16.21 I3C timing register 1 (I3C_TIMINGR1)

Address offset: 0x0A4
Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure some I3C timing settings.

When the I3C acts as target and is controller-capable, this register is used to configure a timing for the controller-role hand-off procedure.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:29</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td><strong>SDA_HD</strong>: SDA hold time (when the I3C acts as controller), in number of kernel clocks cycles (refer to MIPI timing SDA hold time in push-pull t\textsubscript{HD_PP}): SDA hold time = (SDA_HD + 0.5) x t\textsubscript{I3CCLK}</td>
<td></td>
</tr>
<tr>
<td>27:23</td>
<td><strong>FREE[6:0]</strong>: Number of kernel clocks cycles that is used to set some MIPI timings like bus free condition time (when the I3C acts as controller)</td>
<td></td>
</tr>
<tr>
<td>22:16</td>
<td><strong>ASNCR[1:0]</strong></td>
<td></td>
</tr>
<tr>
<td>21:16</td>
<td><strong>AVAL[7:0]</strong></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:29: Reserved, must be kept at reset value.

Bit 28: **SDA\_HD**: SDA hold time (when the I3C acts as controller), in number of kernel clocks cycles (refer to MIPI timing SDA hold time in push-pull t\textsubscript{HD_PP}): SDA hold time = (SDA\_HD + 0.5) x t\textsubscript{I3CCLK}

Note: when controller: t\textsubscript{HD\_PPmin} = min (t\textsubscript{CR}, t\textsubscript{CF}) + 3 ns.

Bits 27:23: Reserved, must be kept at reset value.

Bits 22:16: **FREE[6:0]**: Number of kernel clocks cycles that is used to set some MIPI timings like bus free condition time (when the I3C acts as controller)

When the I3C acts as controller:

1. for I3C start timing: it must wait for (bus free condition) time to be elapsed after a stop and before a start, refer to MIPI timings (I3C) t\textsubscript{CAS} and (I2C) t\textsubscript{BUF}. These timings are defined by: t\textsubscript{BUF} = t\textsubscript{CAS} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA\_HD)] x t\textsubscript{I3CCLK}

Note: For pure I3C bus: t\textsubscript{CASmin} = 38.4 ns, and t\textsubscript{CASmax} = 1 \mu s, 100 \mu s, 2 ms, 50 ms for, respectively, ENTAS0, 1, 2, and 3.

Note: For mixed bus with \textsuperscript{2}C fm\_device t\textsubscript{BUFmin} = 0.5 \mu s, for mixed bus with \textsuperscript{2}C fm device t\textsubscript{BUFmin} = 1.3 \mu s.

2. for I3C repeated start timing: must wait for time to be elapsed after a repeated start (SDA is de-assserted) and before driving SCL low, refer to MIPI timing t\textsubscript{CAS}. This timing is defined by: t\textsubscript{CAS} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA\_HD)] x t\textsubscript{I3CCLK}

Note: t\textsubscript{CASr, min} = 19.2 ns.

3. for I3C stop timing: must wait for time to be elapsed after that the SCL clock is driven high, and before the stop condition (SDA is asserted). This timing is defined by:

\[ t\text{CBP} = (\text{FREE}[6:0] + 1) \times t\text{I3CCLK}. \]

Note: t\textsubscript{CBPmin} = 19.2 ns.

4. for I3C repeated start timing (T-bit when controller ends read with repeated start followed by stop): must wait for time to be elapsed after that the SCL clock is driven high, and before the repeated start condition (SDA is de-assserted). This timing is defined by:

\[ t\text{CBSr} = (\text{FREE}[6:0] + 1) \times t\text{I3CCLK}. \]

Note: t\textsubscript{CBSr, min} = 19.2 ns.

Bits 15:10: Reserved, must be kept at reset value.
Bits 9:8 **ASNCR[1:0]**: Activity state of the new controller (when I3C acts as active controller)  
This field indicates the time to wait before being accessed as new target, refer AVAL[7:0].  
This field can be modified only when the I3C acts as controller.

Bits 7:0 **AVAL[7:0]**: Number of kernel clock cycles to set a time unit of 1 µs, whatever I3C acts as controller or target.  
This field is used by the hardware to build some internal timers, corresponding to the following MIPI 3IC timings:

When the I3C acts as target:
1. for bus available condition time: it must wait for (bus available condition) time to be elapsed after a stop and before issuing a start request for an IBI or a controller-role request (bus free condition is sustained for at least tAVAL). Refer to MIPI timing tAVAL = 1 µs. This timing is defined by: tAVAL = (AVAL[7:0] + 2) x tI3CCLK
2. for bus idle condition time: it must wait for (bus idle condition) time to be elapsed after that both SDA and SCL are continuously high and stable before issuing a hot-join event. Refer to MIPI v1.1 timing tIDLE = 200 µs. This timing is defined by: tIDLE = (AVAL[7:0] + 2) x 200 x tI3CCLK

When the I3C acts as controller, it cannot stall the clock beyond a maximum stall time (stall the SCL clock low), as follows:
1. on first bit of assigned address during dynamic address assignment: it cannot stall the clock beyond the MIPI timing tSTALLDAA = 15 ms. This timing is defined by:  
   tSTALLDAAmax = (AVAL[7:0] + 1) x 15000 x tI3CCLK
2. on ACK/NACK phase of I3C/I2C transfer, on parity bit of write data transfer, on transition bit of I3C read transfer: it cannot stall the clock beyond the MIPI timing tSTALL = 100 µs. This timing is defined by:  
   tSTALLmax = (AVAL[7:0] + 1) x 100 x tI3CCLK

Whatever the I3C acts as controller or as (controller-capable) target, during a controller-role hand-off procedure:
1. The new controller must wait for tNEWCRLock before pulling SDA low (issuing a start) after a completed GETACCR CCC. Then the new controller, within tCAS, can pull SCL low to activate SCL clock. The active controller must wait for the same tNEWCRLock time, or at least 100 µs, before testing if the new controller has gained control of the bus by pulling SDA low. The time to wait depends upon the value of ASNCR[1:0] in the I3C_TIMINGR1 register:
   - ASNCR[1:0] = 00:  
     tNEWCRLock = (AVAL[7:0] + 1) x tI3CCLK
   - ASNCR[1:0] = 01:  
     tNEWCRLock = (AVAL[7:0] + 1) x 100 x tI3CCLK
   - ASNCR[1:0] = 10:  
     tNEWCRLock = (AVAL[7:0] + 1) x 2000 x tI3CCLK
   - ASNCR[1:0] = 11:  
     tNEWCRLock = (AVAL[7:0] + 1) x 50000 x tI3CCLK
49.16.22  I3C timing register 2 (I3C_TIMINGR2)

Address offset: 0x0A8
Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure and enable SCL clock stalling, to enable SCL clock low stalling, if needed by the addressed I3C or legacy I²C target(s).

This register can be written only when the I3C acts as controller.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:8  **STALL[7:0]**: Controller clock stall time, in number of kernel clock cycles

\[ t_{SCLL\_STALL} = STALL \times t_{I3C\_CLK} \]

Bits 7:4  Reserved, must be kept at reset value.

Bit 3  **STALLA**: Controller clock stall enable on ACK phase

The SCL is stalled (during \( t_{SCLL\_STALL} \)) in the address ACK/NACK phase (before the ninth bit). This allows the target to prepare data. The ACK driven by the controller itself on a target-initiated request (IBI/HJ/CR) is not impacted by this control bit.

0: no stall

1: stall enabled

Bit 2  **STALLC**: Controller clock stall enable on PAR phase of CCC

The SCL is stalled during \( STALL \times t_{SCLL\_PP} \) in the T-bit phase of common command code (before the ninth bit). This allows the target to decode the command.

0: no stall

1: stall enabled

*Note: \( t_{SCLL\_PP} = (I3C\_TIMINGR0.SCLL\_PP[7:0] + 1) \times t_{I3C\_CLK} \)*

Bit 1  **STALLD**: Controller clock stall enable on PAR phase of Data

The SCL is stalled during \( STALL \times t_{SCLL\_PP} \) in the T-bit phase (before the ninth bit). This allows the target to read received data.

0: no stall

1: stall enabled

*Note: \( t_{SCLL\_PP} = (I3C\_TIMINGR0.SCLL\_PP[7:0] + 1) \times t_{I3C\_CLK} \)*

Bit 0  **STALLT**: Controller clock stall enable on T-bit phase of data (and on the ACK/NACK phase of data byte of a legacy I²C read)

The SCL is stalled during \( STALL \times t_{SCLL\_PP} \) in the T-bit phase (before the ninth bit). This allows the target to prepare the data to be sent.

0: no stall

1: stall enabled

*Note: \( t_{SCLL\_PP} = (I3C\_TIMINGR0.SCLL\_PP[7:0] + 1) \times t_{I3C\_CLK} \)*
49.16.23 I3C bus characteristics register (I3C_BCR)

Address offset: 0x0C0
Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure three bits used by hardware to return the data byte BCR[7:0] on reception of GETBCR or ENTDAA CCC. The returned byte BCR[7:0] on the I3C bus is then as follows:

- BCR[7] = 0 (reserved)
- BCR[6] = I3C_BCR6 (controller capable)
- BCR[5] = 1 (advanced capabilities, use GETCAPS CCC to determine which ones)
- BCR[4] = 0 (not a virtual target)
- BCR[3] = 1 (offline capable)
- BCR[2] = I3C_BCR2 (IBI payload)
- BCR[1] = 1 (IBI request capable)
- BCR[0] = I3C_BCR0 (max data speed limitation)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

- Bit 6 **BCR6**: Controller capable
  0: I3C target (no controller capable)
  1: I3C controller capable

Bits 5:3 Reserved, must be kept at reset value.

- Bit 2 **BCR2**: in-band interrupt (IBI) payload
  0: no data byte follows the accepted IBI
  1: at least one mandatory data byte follows the accepted IBI (and at most 4 data bytes)

- Bit 1 Reserved, must be kept at reset value.

- Bit 0 **BCR0**: max data speed limitation
  0: no limitation
  1: limitation, as described by I3C_GETMXDSR.
49.16.24 I3C device characteristics register (I3C_DCR)

Address offset: 0x0C4
Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure the device characteristics ID, which is used by hardware to return the data byte DCR[7:0] on reception of GETDCR, ENTDAA, or DEFTGTS CCC.

This register can be written by software only when EN = 0 in the I3C_CFRG register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **DCR[7:0]:** device characteristics ID
0x00: generic device (for v1.0 devices)
others: ID to describe the type of the I3C sensor/device

*Note: The latest MIPI DCR ID assignments are available on https://www.mipi.org.*
### I3C get capability register (I3C_GETCAPR)

- **Address offset:** 0x0C8
- **Reset value:** 0x0000 0000

When the I3C acts as target, this register is used to set the IBI MDB support for pending read notification support, and is used by hardware to return the GETCAP3 byte on reception of GETCAPS CCC, format 1.

The returned byte GETCAP1[7:0] on the I3C bus is then as follows:

- GETCAP1[7:0] = 0 (no HDR)

The returned byte GETCAP2[7:0] on the I3C bus is then as follows:

- GETCAP2[7:6] = 00 (no HDR)
- GETCAP2[5:4] = 00 (no group addressing)
- GETCAP2[3:0] = 0001 (compliant with MIPI specification v1.1)

The returned byte GETCAP3[7:0] on the I3C bus is then as follows:

- GETCAP3[7] = 0 (reserved)
- GETCAP3[6] = CAPPEND in the I3C_GETCAPR register (IBI MDB support for pending read notification)
- GETCAP3[5] = 0 (no HDR)
- GETCAP3[4] = 1 (defining byte support in GETSTATUS)
- GETCAP3[3] = 1 (defining byte support in GETCAPS)
- GETCAP3[2] = 0 (no device-to-device transfer)
- GETCAP3[1] = 0 (no device-to-device transfer)
- GETCAP3[0] = 0 (no multi-lane data transfer)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

<table>
<thead>
<tr>
<th>Bit 31-15</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 14 <strong>CAPPEND</strong>: IBI MDB support for pending read notification</td>
<td></td>
</tr>
<tr>
<td>This bit is written by software during bus initialization (EN = 0), and indicates the support (or not) of the pending read notification via the IBI MDB[7:0] value.</td>
<td></td>
</tr>
<tr>
<td>This bit is used to return the GETCAP3 byte in response to the GETCAPS CCC format 1.</td>
<td></td>
</tr>
<tr>
<td>0: this I3C when acting as target sends an IBI request without a mandatory data byte value indicating a pending read notification</td>
<td></td>
</tr>
<tr>
<td>1: this I3C when acting as target sends an IBI request with a mandatory data byte value (MDB[7:5] = 101), indicating a pending read notification</td>
<td></td>
</tr>
<tr>
<td>Bits 13-0 Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>
49.16.26 I3C controller-role capability register (I3C_CRCAPR)

Address offset: 0x0CC
Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set features the I3C supports as a secondary controller after controller-role hand-off, and is used by hardware to return the CRCAP1 byte and the CRCAP2 byte on reception of GETCAPS CCC, format 2 with the defining byte CRCAPS (0x91).

The returned CRCAP1[7:0] on the I3C bus is then as follows:
- CRCAP1[7:3] = 00000 (reserved)
- CRCAP1[2] = 0 (no multi-lane)
- CRCAP1[1] = CAPGRP in the I3C_CRCAPR register (group management)
- CRCAP1[0] = 1 (hot-join)

The returned CRCAP2[7:0] on the I3C bus is then as follows:
- CRCAP2[7:4] = 0000 (reserved)
- CRCAP2[3] = CAPDHOFF in the I3C_CRCAPR register (delayed controller-role handoff)
- CRCAP2[2] = 1 (deep sleep capable)
- CRCAP2[1] = 0 (no automatic controller-role pass-back)
- CRCAP2[0] = 1 (IBI ack capable)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

<table>
<thead>
<tr>
<th>Bits 31:10</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 9</td>
<td>CAPGRP: group management support (when acting as controller)</td>
</tr>
<tr>
<td></td>
<td>This bit is written by software during bus initialization (EN = 0), and indicates if the I3C is able to support group management when it acts as a controller (after controller-role hand-off) via emitted DEFGRPA, RSTGRPA, and SETGRPA CCC.</td>
</tr>
<tr>
<td></td>
<td>This bit is used to return the CRCAP1 byte in response to the GETCAPS CCC format 2.</td>
</tr>
<tr>
<td>0:</td>
<td>this I3C does not support group address capabilities</td>
</tr>
<tr>
<td>1:</td>
<td>this I3C supports group address capabilities (when becoming controller)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 8:4</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 3</td>
<td>CAPDHOFF: delayed controller-role hand-off</td>
</tr>
<tr>
<td></td>
<td>This bit is written by software during bus initialization (EN = 0), and indicates if this target I3C needs additional time to process a controller-role hand-off requested by the current controller.</td>
</tr>
<tr>
<td></td>
<td>This bit is used to return the CRCAP2 byte in response to the GETCAPS CCC format 2.</td>
</tr>
<tr>
<td>0:</td>
<td>this I3C does not needs additional time to process a controller-role hand-off</td>
</tr>
<tr>
<td>1:</td>
<td>this I3C needs additional time to process a controller-role hand-off</td>
</tr>
</tbody>
</table>
49.16.27 **I3C get max data speed register (I3C_GETMXDSR)**

Address offset: 0x0D0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to return its capabilities and limitations if any. This register is used by hardware to return data byte(s) on reception of GETMXDS CCC, with format 1 (2 bytes, without maxRdTurn), format 2 (5 bytes, with MaxRdTurn), or format 3 (with defining byte WRRDTURN = 0x00: same 5 bytes as format 2, or with defining byte CRHDLY = 0x91: single byte CRHDLY1).

The returned byte MaxWr[7:0] on the I3C bus is then as follows:

- MaxWr[7:4] = 0000 (reserved)
- MaxWr[3] = 1 (defining byte WRRDTURN and CRHDLY support)
- MaxWr[2:0] = 000 (max sustained data rate for non-CCC messages sent by the controller to the target is designed to operate at 12.5 MHz)

The returned byte MaxRd[7:0] on the I3C bus is then as follows:

- MaxRd[7] = 0 (reserved)
- MaxRd[6] = 1 (stop is allowed between write and read)
- MaxRd[5:3] = 100 if TSCO = 0 (clock to data turnaround time tSCO ≤ 12 ns) in the I3C_GETMXDSR register, else 111 (tSCO > 12 ns, refer to the datasheet for more details)
- MaxRd[2:0] = 000 (max sustained data rate for non-CCC messages sent by the target to the controller is designed to operate at 12.5 MHz)

The returned 3-byte MaxRdTurn[23:0], if FMT[1:0] = 00 in the I3C_GETMXDSR register, on the I3C bus is then as follows:

- MaxRdTurn[23:0], with either the MSB (between 65 ms and 16 s), the middle byte (between 256 µs and 65 ms), or the LSB (less than 256 µs) from RDTURN[7:0] in the I3C_GETMXDSR register (others bits are 0), and depending upon FMT[1:0] in the same register.

The returned byte CRHDLY1[7:0] on the I3C bus is then as follows:

- CRHDLY1[7:3] = 00000 (reserved)
- CRHDLY1[2] = 0 if HOFFAS[1:0] = 00 in the I3C_GETMXDSR register, else 1 (set bus activity state)
- CRHDLY1[1:0] = HOFFAS[1:0] (controller-role hand-off activity state)

This register can be written by software only when EN = 0 in the I3C_CFRG register.
Bits 31:25  Reserved, must be kept at reset value.

Bit 24  **TSCO**: clock-to-data turnaround time \(t_{\text{SCO}}\)
This bit is written by software during bus initialization (EN = 0 in the I3C_CFGR register) and is used to specify the clock-to-data turnaround time \(t_{\text{SCO}}\) (vs. the value of 12 ns). This bit is used by the hardware in response to the GETMXDS CCC to return the encoded clock-to-data turnaround time via the returned MaxRd[5:3] bits.
0: \(t_{\text{SCO}} \leq 12\) ns
1: \(t_{\text{SCO}} > 12\) ns (refer to the datasheet for more details)

Bits 23:16  **RDTURN[7:0]**: programmed byte of the 3-byte MaxRdTurn (maximum read turnaround byte)
This bit is written by software during bus initialization (EN = 0) and writes the value of the selected byte (via the FMT[1:0] field) of the 3-byte MaxRdTurn, which is returned in response to the GETMXDS CCC format 2 to encode the maximum read turnaround time.

Bits 15:10  Reserved, must be kept at reset value.

Bits 9:8  **FMT[1:0]**: GETMXDS CCC format
This field is written by software during bus initialization (EN = 0) and indicates how is returned the GETMXDS format 1 (without MaxRdTurn) and format 2 (with MaxRdTurn).
This field is used to return the 2-byte format 1 (MaxWr, MaxRd) or 5-byte format 2 (MaxWr, MaxRd, 3-byte MaxRdTurn) byte in response to the GETCAPS CCC.
00: format 1 (2 bytes with MaxWr with no defining byte, MaxRd)
01: format 2: (5 bytes with MaxWr with no defining byte, MaxRd, MaxRdTurn)
- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = 0 and LSB = RDTURN[7:0].
- Max read turnaround time is less than 256 µs.
10: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, and middle byte of MaxRdTurn)
- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = RDTURN[7:0] and LSB = 0.
- Max read turnaround time is between 256 and 65535 µs
11: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, MSB of MaxRdTurn)
- 3-byte MaxRdTurn is returned with MSB = RDTURN[7:0], middle byte = 0 and LSB = 0.
- Max read turnaround time is between 65535 µs and 16 s.

Bits 7:2  Reserved, must be kept at reset value.

Bits 1:0  **HOFFAS[1:0]**: Controller hand-off activity state
This field is written by software during bus initialization (EN = 0), and indicates in which initial activity state the (other) current controller must expect the I3C bus after a controller-role hand-off to this controller-capable I3C, when returning the defining byte CRHDLY (0x91) to a GETMXDS CCC.
This 2-bit field is used to return the CRHDLY1 byte in response to the GETCAPS CCC format 3, to state the activity state of this I3C when becoming controller after a controller-role hand-off, and consequently the time the former controller must wait before testing this I3C to be confirmed its ownership.
00: activity state 0 is the initial activity state of this I3C before and when becoming controller
01: activity state 1 is the initial activity state of this I3C when becoming controller
10: activity state 2 is the initial activity state of this I3C when becoming controller
11: activity state 3 is the initial activity state of this I3C when becoming controller
49.16.28 I3C extended provisioned ID register (I3C_EPIDR)

Address offset: 0xD4
Reset value: 0x0208 0000

When the I3C acts as target, this register is used to set the 4-bit MIPI instance ID by software, and some other constant bits used for the 48-bit provisioned ID. It is also used by hardware to return the six bytes for the 48-bit provisioned ID on reception of GETPID and ENTDAA CCC.

This register can be written by software only when EN = 0 in the I3C_CFGR register.

|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | I3C_CR        | MEND | MTYPE[3:0] | MIPIMID[14:0] | IDTSEL | Bits 31:17 MIPIMID[14:0]: 15-bit MIPI manufacturer ID
This read field is the 15-bit STMicroelectronics MIPI ID (0x0104). This field represents bits[47:33] of the 48-bit provisioned ID.

Bit 16 IDTSEL: provisioned ID type selector
This field is set as 0 (vendor fixed value). This field represents bit[32] of the 48-bit provisioned ID.

Note: Bits[31:16] of the provisioned ID can be 0.

Bits 15:12 MIPIID[3:0]: 4-bit MIPI Instance ID
This field is written by software to set and identify individually each instance of this I3C IP with a specific number on a single I3C bus. This field represents bits[15:12] of the 48-bit provisioned ID.

Note: Bits[11:0] of the provisioned ID can be 0.

Bits 11:0 Reserved, must be kept at reset value.

49.16.29 I3C register map

Table 529. I3C register map and reset values
| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x004  | I3C_CFGR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x010  | I3C_RDR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x014  | I3C_RDWR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x018  | I3C_TDR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x01C  | I3C_TDWR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x020  | I3C_IBIDR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x024  | I3C_TGTTDR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x030  | I3C_SR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x034  | I3C_SER       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x040  | I3C_RMR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x050  | I3C_EVR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x054  | I3C_IER       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | Reset value | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Table 529. I3C register map and reset values (continued)
## Table 529. I3C register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x058</td>
<td>I3C_CEVR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x05C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x060</td>
<td>I3C_DEVR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x060 + 4 * x, (x = 1 to 4)</td>
<td>I3C_DEVRx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x090</td>
<td>I3C_MAXRLR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x094</td>
<td>I3C_MAXWR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x0A0</td>
<td>I3C_TIMINGR0</td>
<td>SCLH_I2C[7:0]</td>
<td>SCLL_OD[7:0]</td>
<td>SCLH_I3C[7:0]</td>
<td>SCLL_PP[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x0A4</td>
<td>I3C_TIMINGR1</td>
<td>SDA_HD</td>
<td>FREE[6:0]</td>
<td>ASNR[10:0]</td>
<td>AVAL[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x0A8</td>
<td>I3C_TIMINGR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x0C0</td>
<td>I3C_BCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x0C4</td>
<td>I3C_DCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x0C8</td>
<td>I3C_GETCAPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x0CC</td>
<td>I3C_CRCAPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Refer to *Section 2.3 on page 111* for the register boundary addresses.

| Offset  | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x0D0   | I3C_GETMXDSR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0D4   | I3C_EPIDR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | MIPIMID[14:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | MIPIID[3:0]   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value   | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Table 529. I3C register map and reset values (continued)
50 Universal synchronous/asynchronous receiver transmitter (USART/UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART/UART).

50.1 Introduction

The USART offers a flexible means to perform Full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and Half-duplex Single-wire communications, as well as LIN (local interconnection network), Smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and Modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

50.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data
  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous Master/Slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
• Communication control/error detection flags
• Parity control:
  – Transmits parity bit
  – Checks parity of received data byte
• Interrupt sources with flags
• Multiprocessor communications: wake-up from Mute mode by idle line detection or address mark detection
• Wake-up from Stop mode

50.3 **USART extended features**
• LIN master synchronous break send capability and LIN slave break detection capability
  – 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
• IrDA SIR encoder decoder supporting 3/16 bit duration for Normal mode
• Smartcard mode
  – Supports the T=0 and T=1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  – 0.5 and 1.5 stop bits for Smartcard operation
• Support for Modbus communication
  – Timeout feature
  – CR/LF character recognition

50.4 **USART implementation**
The table below describe USART implementation. It also includes LPUART for comparison.

<table>
<thead>
<tr>
<th><strong>Table 530. STM32H563/H573 and STM32H562 features</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>USART modes/features</strong></td>
</tr>
<tr>
<td>USART1</td>
</tr>
<tr>
<td>USART2</td>
</tr>
<tr>
<td>USART3</td>
</tr>
<tr>
<td>USART6</td>
</tr>
<tr>
<td>USART10</td>
</tr>
<tr>
<td>USART11</td>
</tr>
<tr>
<td>UART4</td>
</tr>
<tr>
<td>UART5</td>
</tr>
<tr>
<td>UART7</td>
</tr>
<tr>
<td>UART8</td>
</tr>
<tr>
<td>UART12</td>
</tr>
<tr>
<td>LPUART1</td>
</tr>
</tbody>
</table>
Table 531. USART/LPUART features

<table>
<thead>
<tr>
<th>USART modes/features&lt;sup&gt;(1)&lt;/sup&gt;</th>
<th>Full feature set</th>
<th>Basic feature set</th>
<th>Low-power feature set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware flow control for modem</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Continuous communication using DMA</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Multiprocessor communication</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Synchronous mode (Master/Slave)</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Smartcard mode</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Single-wire Half-duplex communication</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IrDA SIR ENDEC block</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>LIN mode</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Dual clock domain</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Receiver timeout interrupt</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Modbus communication</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Auto baud rate detection</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Driver Enable</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>USART data length</td>
<td>7, 8 and 9 bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tx/Rx FIFO</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Tx/Rx FIFO size</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wake-up from low-power mode</td>
<td>X&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>X&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>X&lt;sup&gt;(2)&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

1. X = supported.
2. Wake-up supported from Stop mode.
50.5 **USART functional description**

50.5.1 **USART block diagram**

![USART block diagram](image)

50.5.2 **USART pins and internal signals**

**Description USART input/output pins**

- **USART bidirectional communications**
  
  USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):
  
  - **RX** (Receive Data Input)
    
    RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.
  
  - **TX** (Transmit Data Output)
    
    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In Single-wire and Smartcard modes, this I/O is used to transmit and receive data.
• RS232 Hardware flow control mode
  The following pins are required in RS232 Hardware flow control mode:
  – **CTS** (Clear To Send)
    When driven high, this signal blocks the data transmission at the end of the current transfer.
  – **RTS** (Request To Send)
    When it is low, this signal indicates that the USART is ready to receive data.

• RS485 Hardware control mode
  The **DE** (Driver Enable) pin is required in RS485 Hardware control mode. This signal activates the Transmission mode of the external transceiver.

• Synchronous Master/Slave mode and Smartcard mode
  The following pins are required in synchronous Master/Slave mode and Smartcard mode:
  – **CK**
    This pin acts as Clock output in Synchronous master and Smartcard modes.
    It acts as Clock input in Synchronous slave mode.
    In Synchronous master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (for example LCD drivers). The clock phase and polarity are software programmable.
    In Smartcard mode, CK output provides the clock to the smartcard.
  – **NSS**
    This pin acts as Slave Select input in Synchronous slave mode.

Refer to [*Table 532*](#) and [*Table 533*](#) for the list of USART input/output pins and internal signals.

#### Table 532. USART input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USART_RX</td>
<td>Input</td>
<td>Serial data receive input.</td>
</tr>
<tr>
<td>USART_TX</td>
<td>Output</td>
<td>Transmit data output.</td>
</tr>
<tr>
<td>USART_CTS</td>
<td>Input</td>
<td>Clear to send</td>
</tr>
<tr>
<td>USART_RTS</td>
<td>Output</td>
<td>Request to send</td>
</tr>
<tr>
<td>USART_DE(^{(1)})</td>
<td>Output</td>
<td>Driver enable</td>
</tr>
<tr>
<td>USART_CK</td>
<td>Output</td>
<td>Clock output in Synchronous master and Smartcard modes.</td>
</tr>
<tr>
<td>USART_NSS(^{(2)})</td>
<td>Input</td>
<td>Slave select input in Synchronous slave mode.</td>
</tr>
</tbody>
</table>

1. USART_DE and USART_RTS share the same pin.
2. USART_NSS and USART_CTS share the same pin.
Description of USART input/output signals

Table 533. USART internal input/output signals

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usart_pclk</td>
<td>Input</td>
<td>APB clock</td>
</tr>
<tr>
<td>usart_ker_ck</td>
<td>Input</td>
<td>USART kernel clock</td>
</tr>
<tr>
<td>usart_wkup</td>
<td>Output</td>
<td>USART provides a wake-up interrupt</td>
</tr>
<tr>
<td>usart_it</td>
<td>Output</td>
<td>USART global interrupt</td>
</tr>
<tr>
<td>usart_tx_dma</td>
<td>Input/output</td>
<td>USART transmit DMA request</td>
</tr>
<tr>
<td>usart_rx_dma</td>
<td>Input/output</td>
<td>USART receive DMA request</td>
</tr>
</tbody>
</table>

50.5.3 USART clocks

The simplified block diagram given in Figure 674 shows two fully-independent clock domains:

- The **usart_pclk** clock domain
  
  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.

- The **usart_ker_ck** kernel clock domain.
  
  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.
  
  When the dual clock domain feature is not supported, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

50.5.4 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see Figure 675):

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

**Note:** In 7-bit data length mode, the Smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.
An **Idle character** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A **Break character** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.
9-bit word length (M = 01), 1 Stop bit

Data frame

9-bit word length (M = 01), 1 Stop bit

Possible Parity bit

Next

Start bit

Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Bit8 Stop bit

Idle frame

Break frame

Start bit

MS33194V2

Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Stop bit

Idle frame

Break frame

MS33194V2

8-bit word length (M = 00), 1 Stop bit

Data frame

8-bit word length (M = 00), 1 Stop bit

Possible Parity bit

Next

Start bit

Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Stop bit

Idle frame

Break frame

Start bit

MS33194V2

Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Stop bit

Idle frame

Break frame

MS33194V2

7-bit word length (M = 10), 1 Stop bit

Data frame

7-bit word length (M = 10), 1 Stop bit

Possible Parity bit

Next

Start bit

Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Stop bit

Idle frame

Break frame

Start bit

MS33194V2

Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Stop bit

Idle frame

Break frame

MS33194V2

** LBCL bit controls last data clock pulse
50.5.5 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and Smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in USART_CR3 control register.

In this case:
- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.
  
  In this case, the RXFT flag is set in the USART_ISR register. This means that RXFTCFG data have been received: 1 data in USART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size -1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data does not set the overrun flag.

- The Tx interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

50.5.6 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.
Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR. The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost. An idle frame is sent when the TE bit is enabled.

Configurable stop bits

The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.
- **1 stop bit**: This is the default value of number of stop bits.
- **2 stop bits**: This is supported by normal USART, Single-wire and Modem modes.
- **1.5 stop bits**: To be used in Smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when \(M[1:0] = 00\)) or 11 low bits (when \(M[1:0] = 01\)) or 9 low bits (when \(M[1:0] = 10\)) followed by 2 stop bits (see Figure 676). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

![Figure 676. Configurable stop bits](image-url)

Character transmission procedure

To transmit a character, follow the sequence below:
1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in Section 50.5.20: Continuous communication using USART and DMA.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.

8. When the last data is written to the USART_TDR register, wait until TC=1.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.
This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

Single byte communication

• When FIFO mode is disabled
  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:
  – the data have been moved from the USART_TDR register to the shift register and the data transmission has started;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data.
This flag generates an interrupt if the TXEIE bit is set.
  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.
  When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

• When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – the TXFIFO is not full;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.
Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.
If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.
After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the microcontroller to enter the low-power mode (see Figure 677: TC/TXE behavior when transmitting).

Figure 677. TC/TXE behavior when transmitting

Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see Figure 675).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

50.5.7 USART receiver

The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.
In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0 X 0X 0.

Figure 678. Start bit detection when oversampling by 16 or 8

Note: If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE=1, or RXFNE flag set and interrupt generated if RXFNEIE=1 if FIFO mode enabled) if the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated but the NE noise flag is set if,
   a) for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)
   or
   b) for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).
Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in Section 50.5.20: Continuous communication using USART and DMA.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In Multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty i.e. when there are data to be read from the RXFIFO.
- In Single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to 1 in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to 1 in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be
generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

**Break character**

When a break character is received, the USART handles it as a framing error.

**Idle character**

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

**Overrun error**

- **FIFO mode disabled**
  
  An overrun error occurs if a character is received and RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte reception. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:
  - the ORE bit is set;
  - the RDR content is not lost. The previous data is available by reading the USART_RDR register.
  - the shift register is overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXNEIE or the EIE bit is set.

- **FIFO mode enabled**
  
  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full. Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty. An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:
  - The ORE bit is set.
  - The first entry in the RXFIFO is not lost. It is available by reading the USART_RDR register.
  - The shift register is overwritten. After that point, any data received during overrun is lost.
  - An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECFF bit in the USART_ICR register.

*Note:* The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- if RXNE=1, then the last valid data is stored in the receive register (RDR) and can be read,
- if RXNE=0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.
Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see Section: Reset and Clock Control (RCC)). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is usart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the usart_ker_ck clock source can be configurable in the RCC (see Section: Reset and Clock Control (RCC)). Otherwise the usart_ker_ck clock is the same as usart_pclk.

The usart_ker_ck clock can be divided by a programmable factor, defined in the USART_PRES register.

Some usart_ker_ck sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in Synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (see Figure 680 and Figure 681).

Depending on the application:

- select oversampling by 8 (OVER8=1) to achieve higher speed (up to usart_ker_ck_pres/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to Section 50.5.9: Tolerance of the USART receiver to clock deviation on page 2248)
- select oversampling by 16 (OVER8=0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum
usr

The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.

• A single sample in the center of the received bit

Depending on the application:

– select the three sample majority vote method (ONEBIT=0) when operating in a noisy environment and reject the data when a noise is detected (refer to Table 534) because this indicates that a glitch occurred during the sampling.

– select the single sample method (ONEBIT=1) when the line is noise-free to increase the receiver tolerance to clock deviations (see Section 50.5.9: Tolerance of the USART receiver to clock deviation on page 2248). In this case the NE bit is never set.

When noise is detected in a frame:

• The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).

• The invalid data is transferred from the Shift register to the USART_RDR register.

• No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting NFCF bit in ICR register.

Note: Noise error is not supported in SPI mode.

Oversampling by 8 is not available in the Smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to 0 by hardware.

Figure 680. Data sampling when oversampling by 16
Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.
Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in Normal mode and 0.5 or 1.5 in Smartcard mode.

- **0.5 stop bit (reception in Smartcard mode):** no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit:** sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **1.5 stop bits (Smartcard mode)**
  When transmitting in Smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE =1 in USART_CR1) and the stop bit is checked to test if the Smartcard has detected a parity error.
  
  In the event of a parity error, the Smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to Section 50.5.17: USART receiver timeout on page 2261 for more details).
  
  - **2 stop bits**
    Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit. The framing error flag is set if a framing error is detected during the first stop bit. The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

**50.5.8 USART baud rate generation**

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

**Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = 0 or 1)**

In case of oversampling by 16, the baud rate is given by the following formula:

\[
\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}
\]

In case of oversampling by 8, the baud rate is given by the following formula:

\[
\text{Tx/Rx baud} = \frac{2 \times \text{usart_ker_ck_pres}}{\text{USARTDIV}}
\]

**Equation 2: baud rate in Smartcard, LIN and IrDA modes (OVER8 = 0)**

The baud rate is given by the following formula:

\[
\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}
\]
USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When $\text{OVER8} = 0$, $\text{BRR} = \text{USARTDIV}$.
- When $\text{OVER8} = 1$
  - $\text{BRR}[2:0] = \text{USARTDIV}[3:0]$ shifted 1 bit to the right.
  - $\text{BRR}[3]$ must be kept cleared.
  - $\text{BRR}[15:4] = \text{USARTDIV}[15:4]$

**Note:** The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value must not be changed during communication.

*In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.*

**How to derive USARTDIV from USART_BRR register values**

**Example 1**
To obtain 9600 bauds with $\text{usart\_ker\_ck\_pres} = 8$ MHz:

- In case of oversampling by 16:
  - $\text{USARTDIV} = 8\,000\,000/9600$
  - $\text{BRR} = \text{USARTDIV} = 0\text{d}833 = 0x0341$

- In case of oversampling by 8:
  - $\text{USARTDIV} = 2 \times 8\,000\,000/9600$
  - $\text{USARTDIV} = 1666.66$ (0d1667 = 0x683)
  - $\text{BRR}[3:0] = 0x3 >> 1 = 0x1$
  - $\text{BRR} = 0x681$

**Example 2**
To obtain 921.6 kbauds with $\text{usart\_ker\_ck\_pres} = 48$ MHz:

- In case of oversampling by 16:
  - $\text{USARTDIV} = 48\,000\,000/921\,600$
  - $\text{BRR} = \text{USARTDIV} = 0\text{x}52 = 0x34$

- In case of oversampling by 8:
  - $\text{USARTDIV} = 2 \times 48\,000\,000/921\,600$
  - $\text{USARTDIV} = 104$ (0d104 = 0x68)
  - $\text{BRR}[3:0] = \text{USARTDIV}[3:0] >> 1 = 0x8 >> 1 = 0x4$
  - $\text{BRR} = 0x64$
50.5.9 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter’s local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

\[ \text{DTRA} + \text{DQUANT} + \text{DREC} + \text{DTCL} + \text{DWU} < \text{USART receiver tolerance} \]

where

- DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when \( M[1:0] = 01 \):

\[ \text{DWU} = \frac{t_{\text{WUUSART}}}{11 \times \text{Tbit}} \]

when \( M[1:0] = 00 \):

\[ \text{DWU} = \frac{t_{\text{WUUSART}}}{10 \times \text{Tbit}} \]

when \( M[1:0] = 10 \):

\[ \text{DWU} = \frac{t_{\text{WUUSART}}}{9 \times \text{Tbit}} \]

\( t_{\text{WUUSART}} \) is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in Table 535, Table 536, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.
Note: The data specified in Table 535 and Table 536 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M= 01 or 9-bit times when M = 10).

### 50.5.10 USART auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from `usart_ker_ck_pres/65535` and `usart_ker_ck_pres/16`.
- When oversampling by 8, the baud rate ranges from `usart_ker_ck_pres/65535` and `usart_ker_ck_pres/8`.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.
These modes are the following:

- **Mode 0**: Any character starting with a bit at 1.
  In this case the USART measures the duration of the start bit (falling edge to rising edge).

- **Mode 1**: Any character starting with a 10xx bit pattern.
  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.

- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).
  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.

- **Mode 3**: A 0x55 character frame.
  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit 6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a 0). When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection must be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

*Note:* The BRR value might be corrupted if the USART is disabled (UE=0) during an auto baud rate operation.
**50.5.11 USART multiprocessor communication**

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the USART_CR1 register.

*Note:* When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise Mute mode might remain active.

When the Mute mode is enabled:
- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

**Idle line detection (WAKE=0)**

The USART enters Mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of Mute mode behavior using Idle line detection is given in *Figure 682.*
Figure 682. Mute mode using Idle line detection

Note: If the MMRQ is set while the IDLE character has already elapsed, Mute mode is not entered (RWU is not set).
If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a 1, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters Mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters Mute mode. When FIFO management is enabled, the software must ensure that there is at least one empty location in the RXFIFO before entering Mute mode.

The USART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from Mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in Mute mode

An example of Mute mode behavior using address mark detection is given in Figure 683.
50.5.12 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a Half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.
50.5.13 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in Table 537.

<table>
<thead>
<tr>
<th>M bits</th>
<th>PCE bit</th>
<th>USART frame(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).
50.5.14 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to Section 50.4: USART implementation on page 2229.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:
- CLKEN in the USART_CR2 register,
- STOP[1:0], SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in Section 50.5.5 has to be applied for LIN Master transmission. It must be the same as for normal USART transmission with the following differences:
- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 zero bits as a break character. Then 2 bits of value ‘1’ are sent to enable the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE=1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL=1 in USART_CR2) consecutive bits are detected as 0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a ‘1’ is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (i.e. stop bit detected at 0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1’, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the Figure 684: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 2256.

Examples of break frames are given on Figure 685: Break detection in LIN mode vs. Framing error detection on page 2257.
**Figure 684. Break detection in LIN mode (11-bit break length - LBDL bit is set)**

**Case 1: break signal not long enough => break discarded, LBDF is not set**

<table>
<thead>
<tr>
<th>RX line</th>
<th>Break frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture strobe</td>
<td>Break state machine</td>
</tr>
<tr>
<td></td>
<td>Idle Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Bit8 Bit9 Bit10 Idle</td>
</tr>
<tr>
<td>Read samples</td>
<td>0 0 0 0 0 0 0 0 0 1</td>
</tr>
</tbody>
</table>

**Case 2: break signal just long enough => break detected, LBDF is set**

<table>
<thead>
<tr>
<th>RX line</th>
<th>Break frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture strobe</td>
<td>Break state machine</td>
</tr>
<tr>
<td></td>
<td>Idle Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Bit8 Bit9 Bit10 Idle</td>
</tr>
<tr>
<td>Read samples</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>LBDF</td>
<td>wait delimiter</td>
</tr>
</tbody>
</table>

**Case 3: break signal long enough => break detected, LBDF is set**

<table>
<thead>
<tr>
<th>RX line</th>
<th>Break frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture strobe</td>
<td>Break state machine</td>
</tr>
<tr>
<td></td>
<td>Idle Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Bit8 Bit9 Bit10 wait delimiter Idle</td>
</tr>
<tr>
<td>Read samples</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>LBDF</td>
<td></td>
</tr>
</tbody>
</table>
50.5.15 **USART synchronous mode**

**Master mode**

The Synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to 1. In Synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in Master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see Figure 686, Figure 687 and Figure 688).

During the Idle state, preamble and send break, the external CK clock is not activated.

In Synchronous master mode, the USART transmitter operates exactly like in Asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In Synchronous master mode, the USART receiver operates in a different way compared to Asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).
Note: In Master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 686. USART example of synchronous master transmission

Figure 687. USART data clock timing diagram in Synchronous master mode (M bits =00)
Figure 688. USART data clock timing diagram in Synchronous master mode (M bits = 01)

Slave mode

The Synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to 1. In Synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in Slave mode. The CK pin is the input of the USART in Slave mode.

Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (usart_ker_ck_pres) must be greater than 3 times the CK input frequency.

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see Figure 689).

An underrun error flag is available in Slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.
Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- Software NSS management (DIS_NSS = 1)
  SPI slave is always selected and NSS input pin is ignored.
  The external NSS pin remains free for other application uses.

- Hardware NSS management (DIS_NSS = 0)
  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE=0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master.

The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

SPI Slave underrun error

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.
In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

**Note:** An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART_TDR must be written 3 usec before the first CK edge.

### 50.5.16 USART single-wire Half-duplex communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

### 50.5.17 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = 00 or STOP = 11
- from the end of the second stop bit if STOP = 10.
- from the beginning of the stop bit if STOP = 01.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.
50.5.18 USART Smartcard mode

This section is relevant only when Smartcard mode is supported. Refer to Section 50.4: USART implementation on page 2229.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In Smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the Smartcard.

The Smartcard interface is designed to support asynchronous Smartcard protocol as defined in the ISO 7816-3 standard. Both T=0 (character mode) and T=1 (block mode) are supported.

The USART must be configured as:

- 8 bits plus parity: M=1 and PCE=1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP=11 in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T=0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 690 shows examples of what can be seen on the data line with and without parity error.

![Figure 690. ISO 7816-3 asynchronous protocol](image)

When connected to a Smartcard, the TX output of the USART drives a bidirectional line that is also driven by the Smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.

- In transmission, if the Smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.
The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.

- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the Smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T=1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the Smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.

- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn’t repeat the character.

- In transmission, the USART inserts the Guard Time (as programmed in the Guard Time register) between two successive characters. As the Guard Time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT (Character Guard Time, as defined by the 7816-3 specification) minus 12 (the duration of one character).

- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the Guard Time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the Guard Time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.

- The de-assertion of TC flag is unaffected by Smartcard mode.

- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.

- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: Break characters are not significant in Smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 691 shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.
The USART can provide a clock to the Smartcard through the CK output. In Smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. CK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

**Block mode (T=1)**

In T=1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the UART_CR3 register.

When requesting a read from the Smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) - 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

**Note:** The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the Smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time -11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the Smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

**Note:** As in the Smartcard protocol definition, the BWT/CWT values must be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT -11 or CWT -11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the Smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value.
(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The Smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=0, DATAINV=0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=1, DATAINV=1.

Note: When logical data values are inverted (0=H, 1=L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to ‘3F’.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to ‘3B’.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH => the USART received character is equal to 03 and the parity is odd.
Therefore, two methods are available for TS pattern recognition:

**Method 1**
The USART is programmed in standard Smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.
- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

**Method 2**
The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:
- (H) LHHL LLL LLH = 0x103 -> inverse convention to be chosen
- (H) LHHL HHH LLH = 0x13B -> direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

### 50.5.19 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 50.4: USART implementation on page 2229*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:
- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 692*).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 kbauds for the SIR ENDEC. In Normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.
- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not
encoded. While receiving data, transmission must be avoided as the data to be transmitted may be corrupted.

- A 0 is transmitted as a high pulse and a 1 is transmitted as a 0. The width of the pulse is specified as 3/16th of the selected bit period in Normal mode (see Figure 693).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods are accepted as a pulse. The IrDA encoder/decoder doesn't work when PSC=0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to ‘1 stop bit’.

**IrDA low-power mode**

- **Transmitter**
  In low-power mode, the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < PSC< 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.

- **Receiver**
  Receiving in low-power mode is similar to receiving in Normal mode. For glitch detection the USART must discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power baud clock (PSC value in the USART_GTPR).

**Note:**
A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.

The receiver set up time must be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).
Figure 692. IrDA SIR ENDEC block diagram

Figure 693. IrDA data modulation (3/16) - Normal mode
50.5.20 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 50.4: USART implementation on page 2229 to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 50.5.7. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to section Direct memory access controller (DMA)) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In Transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.
**Reception using DMA**

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to *section Direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register.
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

**Note:** *The DMAR bit must not be cleared before the DMA end of transfer.*
Figure 695. Reception using DMA

<table>
<thead>
<tr>
<th>TX line</th>
<th>Frame 1</th>
<th>Frame 2</th>
<th>Frame 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXNE flag</td>
<td>Set by hardware</td>
<td>cleared by DMA read</td>
<td>cleared by software</td>
</tr>
<tr>
<td>DMA request</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USART_RDR</td>
<td>F1</td>
<td>F2</td>
<td>F3</td>
</tr>
<tr>
<td>DMA reads USART_RDR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMA TCIF flag (transfer complete)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Software configures the DMA to receive 3 data blocks and enables the USART

Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in Multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

50.5.21 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The Figure 696 shows how to connect 2 devices in this mode:

Figure 696. Hardware flow control between 2 USARTs

Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).
RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to 1 in the USART_CR3 register.

**RS232 RTS flow control**

If the RTS flow control is enabled (RTSE=1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 697* shows an example of communication with RTS flow control enabled.

*Figure 697. RS232 RTS flow control*

![RS232 RTS flow control diagram](image)

**Note:** When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

**RS232 CTS flow control**

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 698* shows an example of communication with CTS flow control enabled.
For correct behavior, CTS must be deasserted at least 3 USART clock source periods before the end of the current character. In addition it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

**RS485 driver enable**

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the USART_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).
50.5.22 USART low-power management

The USART has advanced low-power mode functions, that enables transferring properly data even when the usart_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set. When the usart_pclk is gated, the USART provides a wake-up interrupt (USART_wkup) if a specific action requiring the activation of the usart_pclk clock is needed:

- If FIFO mode is disabled
  usart_pclk clock has to be activated to empty the USART data register.
  In this case, the USART interrupt source is RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.

- If FIFO mode is enabled
  usart_pclk clock has to be activated to:
  - to fill the TXFIFO
  - or to empty the RXFIFO
  In this case, the USART interrupt source can be:
    - RXFIFO not empty. In this case, the RXFIEE bit must be set before entering low-power mode.
    - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
    - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

To avoid overrun/underrun errors and transmit/receive data in low-power mode, the USART interrupt source can be one of the following events:

- TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
- RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time required to receive a single byte across the line.

Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific USART_wkup interrupt can be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and a USART_wkup interrupt is generated if the WUFIE bit is set.
Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or Active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled when exiting from low-power mode.

When the FIFO is enabled, waking up from low-power mode on address match is only possible when Mute mode is enabled.

**Using Mute mode with low-power mode**

If the USART is put into Mute mode before entering low-power mode:

- Wake-up from Mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from Mute mode on address match is used, then the low-power mode wake-up source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in Mute mode upon address match and wake up from low-power mode.

**Note:** When FIFO management is enabled, Mute mode can be used with wake-up from low-power mode without any constraints (i.e. the two points mentioned above about Mute and low-power mode are valid only when FIFO management is disabled).

**Wake-up from low-power mode when USART kernel clock (usart_ker_ck) is OFF in low-power mode**

If during low-power mode, the usart_ker_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart_ker_ck clock to be switched ON thanks to the usart_ker_ck_req signal. usart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, usart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.
Figure 699 shows the USART behavior when the wake-up event is verified.

Figure 699. Wake-up event verified (wake-up event = address match, FIFO disabled)

Figure 700 shows the USART behavior when the wake-up event is not verified.

Figure 700. Wake-up event not verified (wake-up event = address match, FIFO disabled)

Note: The figures above are valid when address match or any received frame is used as wake-up event. If the wake-up event is the start bit detection, the USART sends the wake-up event to the MCU at the end of the start bit.
Determining the maximum USART baud rate that enables to correctly wake up the microcontroller from low-power mode

The maximum baud rate that enables to correctly wake up the microcontroller from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see Section 50.5.9: Tolerance of the USART receiver to clock deviation).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR[3:0] = 0000.

In these conditions, according to Table 535: Tolerance of the USART receiver when BRR[3:0] = 0000, the USART receiver tolerance equals 3.41%.

\[
D_{WUmax} = \frac{t_{WUUSART}}{11 \times T_{bitmin}}
\]

\[
T_{bitmin} = \frac{t_{WUUSART}}{11 \times D_{WUmax}}
\]

where \(t_{WUUSART}\) is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In fact, we need to consider at least the \(\text{uart}_\text{ker}_\text{ck}\) inaccuracy (DREC).

For example, if HSI is used as \(\text{uart}_\text{ker}_\text{ck}\), and the HSI inaccuracy is of 1%, then we obtain:

\[
D_{WUmax} = \text{USART receiver tolerance} - \text{DREC} = 3.41\% - 1\% = 2.41\%
\]

\[
T_{bitmin} = \frac{3 \mu s}{11 \times 2.41\%} = 11.32 \mu s.
\]

As a result, the maximum baud rate enables to wake up correctly from low-power mode is: \(1/11.32 \mu s = 88.36\) kbauds.

50.6 USART in low-power modes

Table 538. Effect of low-power modes on the USART

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. USART interrupts cause the device to exit Sleep mode.</td>
</tr>
<tr>
<td>Stop(1)</td>
<td>The content of the USART registers is kept. The USART is able to wake up the microcontroller from Stop mode when the USART is clocked by an oscillator available in Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The USART peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

1. Refer to Section 50.4: USART implementation to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

50.7 USART interrupts

Refer to Table 539 for a detailed description of all USART interrupt requests.
<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop(1) modes</th>
<th>Exit from Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>USART or UART</td>
<td>Transmit data register empty</td>
<td>TXE</td>
<td>TXEIE</td>
<td>Write TDR</td>
<td>No</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transmit FIFO Not Full</td>
<td>TXFNF</td>
<td>TXFNFIE</td>
<td>TXFIFO full</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transmit FIFO Empty</td>
<td>TXFE</td>
<td>TXFEIE</td>
<td>Write TDR or write 1 in TXFRQ</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transmit FIFO threshold reached</td>
<td>TXFT</td>
<td>TXFTIE</td>
<td>Write TDR</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CTS interrupt</td>
<td>CTSIF</td>
<td>CTSIE</td>
<td>Write 1 in CTSCF</td>
<td>No</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transmission Complete</td>
<td>TC</td>
<td>TCIE</td>
<td>Write TDR or write 1 in TCCF</td>
<td>No</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transmission Complete Before Guard Time</td>
<td>TCBGT</td>
<td>TCBGTIE</td>
<td>Write TDR or write 1 in TCBGT</td>
<td>No</td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>
Table 539. USART interrupt requests (continued)

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop(1) modes</th>
<th>Exit from Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Receive data register not empty (data ready to be read)</td>
<td>RXNE</td>
<td>RXNEIE</td>
<td>Read RDR or write 1 in RXFRQ</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Receive FIFO Not Empty</td>
<td>RXFNE</td>
<td>RXFNEIE</td>
<td>Read RDR until RXFIFO empty or write 1 in RXFRQ</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Receive FIFO Full</td>
<td>RXFF(2)</td>
<td>RXFFIE</td>
<td>Read RDR</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Receive FIFO threshold reached</td>
<td>RXFT</td>
<td>RXFTIE</td>
<td>Read RDR</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overrun error detected</td>
<td>ORE</td>
<td>RXNEIE/RXFNEIE</td>
<td>Write 1 in ORECF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Idle line detected</td>
<td>IDLE</td>
<td>IDLEIE</td>
<td>Write 1 in IDLECF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Parity error</td>
<td>PE</td>
<td>PEIE</td>
<td>Write 1 in PECF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LIN break</td>
<td>LBDF</td>
<td>LBDIE</td>
<td>Write 1 in LBDCF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Noise error in multibuffer communication.</td>
<td>NE</td>
<td></td>
<td>Write 1 in NFCF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overrun error in multibuffer communication.</td>
<td>ORE(3)</td>
<td>EIE</td>
<td>Write 1 in ORECF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Framing Error in multibuffer communication.</td>
<td>FE</td>
<td></td>
<td>Write 1 in FECF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Character match</td>
<td>CMF</td>
<td>CMIE</td>
<td>Write 1 in CMCF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Receiver timeout</td>
<td>RTOF</td>
<td>RTOFIE</td>
<td>Write 1 in RTOCCF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>End of Block</td>
<td>EOBF</td>
<td>EOBIE</td>
<td>Write 1 in EOBCF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wake-up from low-power mode</td>
<td>WUF</td>
<td>WUFIE</td>
<td>Write 1 in WUC</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPI slave underrun error</td>
<td>UDR</td>
<td>EIE</td>
<td>Write 1 in UDRCF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to Section 50.4: USART implementation for the list of supported Stop modes.

2. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).

3. When OVRDIS = 0.
50.8 USART registers

Refer to Section 1.2 on page 101 for a list of abbreviations used in register descriptions.
The peripheral registers have to be accessed by words (32 bits).

50.8.1 USART control register 1 (USART_CR1)

Address offset: 0x00
Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXF FIE</td>
<td>TXFEIE</td>
<td>FIFO EN</td>
<td>M1</td>
<td>EOBIE</td>
<td>RTOIE</td>
<td>DEAT[4:0]</td>
<td>DEDT[4:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 RXFFIE: RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when RXFF=1 in the USART_ISR register

Bit 30 TXFEIE: TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFE=1 in the USART_ISR register

Bit 29 FIFOEN: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE=0).

Note: FIFO mode can be used on standard UART communication, in SPI Master/Slave mode and in Smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 M1: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 start bit, 8 Data bits, n Stop bit
M[1:0] = 01: 1 start bit, 9 Data bits, n Stop bit
M[1:0] = 10: 1 start bit, 7 Data bits, n Stop bit
This bit can only be written when the USART is disabled (UE=0).

Note: In 7-bits data length mode, the Smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.
Bit 27 **EOBIE**: End of Block interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the EOBF flag is set in the USART_ISR register
*Note*: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 26 **RTOIE**: Receiver timeout interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.
*Note*: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 50.4: USART implementation on page 2229.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time
This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).
This bitfield can only be written when the USART is disabled (UE=0).
*Note*: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time
This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).
If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DDT and DEAT times have both elapsed.
This bitfield can only be written when the USART is disabled (UE=0).
*Note*: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 15 **OVER8**: Oversampling mode
0: Oversampling by 16
1: Oversampling by 8
This bit can only be written when the USART is disabled (UE=0).
*Note*: In LIN, IrDA and Smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable
This bit enables the USART Mute mode function. When set, the USART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.
0: Receiver in Active mode permanently
1: Receiver can switch between Mute mode and Active mode.

Bit 12 **M0**: Word length
This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).
This bit can only be written when the USART is disabled (UE=0).

---

**RM0481 Universal synchronous/asynchronous receiver transmitter (USART/UART)**
Bit 11 **WAKE**: Receiver wake-up method
This bit determines the USART wake-up method from Mute mode. It is set or cleared by software.
0: Idle line
1: Address mark
This bitfield can only be written when the USART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled
This bitfield can only be written when the USART is disabled (UE=0).

Bit 9 **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the USART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever PE=1 in the USART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TXFNF =1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TC=1 in the USART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever ORE=1 or RXFNE=1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever IDLE=1 in the USART_ISR register
Bit 3  **TE:** Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled
1: Transmitter is enabled

*Note:* During transmission, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2  **RE:** Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1  **UESM:** USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.
1: USART able to wake up the MCU from low-power mode.

*Note:* It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0  **UE:** USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode
1: USART enabled

*Note:* To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In Smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.
50.8.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00
Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Access</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FIFOEN</td>
<td>rw</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>M1</td>
<td>rw</td>
<td>M[1:0] = 00</td>
</tr>
<tr>
<td>29</td>
<td>EOBIE</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>RTOIE</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>DEAT[4:0]</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>DEDT[4:0]</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>OVER8</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>CMIE</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>MME</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>M0</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>WAKE</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>PCE</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>PS</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>PEIE</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>TXIE</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>TCIE</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>RXNEIE</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>IDLEIE</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>TE</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>RE</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>UESM</td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>UE</td>
<td>rw</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 FIFOEN: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

This bitfield can only be written when the USART is disabled (UE=0).

Note: FIFO mode can be used on standard UART communication, in SPI Master/Slave mode and in Smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 M1: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 start bit, 8 Data bits, n Stop bit
M[1:0] = 01: 1 start bit, 9 Data bits, n Stop bit
M[1:0] = 10: 1 start bit, 7 Data bits, n Stop bit

This bit can only be written when the USART is disabled (UE=0).

Note: In 7-bits data length mode, the Smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 EOBIE: End of Block interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 26 RTOIE: Receiver timeout interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 50.4: USART implementation on page 2229.
Bits 25:21 \textbf{DEAT[4:0]}: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE=0).

\textit{Note:} If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bits 20:16 \textbf{DEDT[4:0]}: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE=0).

\textit{Note:} If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 15 \textbf{OVER8}: Oversampling mode

0: Oversampling by 16
1: Oversampling by 8

This bit can only be written when the USART is disabled (UE=0).

\textit{Note:} In LIN, IrDA and Smartcard modes, this bit must be kept cleared.

Bit 14 \textbf{CMIE}: Character match interrupt enable

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 \textbf{MME}: Mute mode enable

This bit enables the USART Mute mode function. When set, the USART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.
0: Receiver in Active mode permanently
1: Receiver can switch between Mute mode and Active mode.

Bit 12 \textbf{M0}: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the USART is disabled (UE=0).

Bit 11 \textbf{WAKE}: Receiver wake-up method

This bit determines the USART wake-up method from Mute mode. It is set or cleared by software.
0: Idle line
1: Address mark

This bitfield can only be written when the USART is disabled (UE=0).

Bit 10 \textbf{PCE}: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if \(M=1\); 8th bit if \(M=0\)) and the parity is checked on the received data. This bit is set and cleared by software.

Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE=0).
Bit 9 **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the USART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever PE=1 in the USART_ISR register

Bit 7 **TXEIE**: Transmit data register empty
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TXE =1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TC=1 in the USART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever ORE=1 or RXNE=1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever IDLE=1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled

**Note:** During transmission, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.
Bit 2 **RE**: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode
When this bit is cleared, the USART cannot wake up the MCU from low-power mode.
When this bit is set, the USART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: USART not able to wake up the MCU from low-power mode.
1: USART able to wake up the MCU from low-power mode.

*Note*: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: USART enable
When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.
0: USART prescaler and outputs disabled, low-power mode
1: USART enabled

*Note*: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.
In Smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

### 50.8.3 USART control register 2 (USART_CR2)
Address offset: 0x04
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD[7:0]</td>
<td>RTOEN</td>
<td>ABRMOD[1:0]</td>
<td>ABREN</td>
<td>MSBF</td>
<td>RST</td>
<td>DATAINV</td>
<td>TXINV</td>
<td>RXINV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SWAP</td>
<td>LINEN</td>
<td>STOP[1:0]</td>
<td>CLKEN</td>
<td>CPOL</td>
<td>CPHA</td>
<td>LBCL</td>
<td>Res.</td>
<td>LBDIE</td>
<td>LBBL</td>
<td>ADDM7</td>
<td>DIS NSS</td>
<td>Res</td>
<td>Res</td>
<td>SLVEN</td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:24  **ADD[7:0]: Address of the USART node**

These bits give the address of the USART node in Mute mode or a character code to be recognized in low-power or Run mode:

- In Mute mode: they are used in multiprocessor communication to wake up from Mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- In Run mode with Mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the USART is disabled (UE = 0).

Bit 23  **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.
1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

**Note**: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bits 22:21  **ABRMOD[1:0]: Auto baud rate mode**

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.
01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)
10: 0x7F frame detection.
11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE=0).

**Note**: If DATAINV=1 and/or MSBFIRST=1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 20  **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.
1: Auto baud rate detection is enabled.

**Note**: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 19  **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE=0).
Bit 18 **DATAINV**: Binary data inversion
This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
This bitfield can only be written when the USART is disabled (UE=0).

Bit 17 **TXINV**: TX pin active level inversion
This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (VDD =1/idle, Gnd=0/mark)
1: TX pin signal values are inverted. (VDD =0/mark, Gnd=1/idle).
This enables the use of an external inverter on the TX line.
This bitfield can only be written when the USART is disabled (UE=0).

Bit 16 **RXINV**: RX pin active level inversion
This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (VDD =1/idle, Gnd=0/mark)
1: RX pin signal values are inverted. (VDD =0/mark, Gnd=1/idle).
This enables the use of an external inverter on the RX line.
This bitfield can only be written when the USART is disabled (UE=0).

Bit 15 **SWAP**: Swap TX/RX pins
This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.
This bitfield can only be written when the USART is disabled (UE=0).

Bit 14 **LINEN**: LIN mode enable
This bit is set and cleared by software.
0: LIN mode disabled
1: LIN mode enabled
The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.
This bitfield can only be written when the USART is disabled (UE=0).

*Note*: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value.
Refer to Section 50.4: USART implementation on page 2229.

Bits 13:12 **STOP[1:0]**: stop bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: 0.5 stop bit.
10: 2 stop bits
11: 1.5 stop bits
This bitfield can only be written when the USART is disabled (UE=0).
Bit 11 **CLKEN**: Clock enable
This bit enables the user to enable the CK pin.
0: CK pin disabled
1: CK pin enabled
This bit can only be written when the USART is disabled (UE=0).

**Note:** If neither Synchronous mode nor Smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

In Smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:

- **UE** = 0
- **SCEN** = 1
- GTPR configuration
- **CLKEN** = 1
- **UE** = 1

Bit 10 **CPOL**: Clock polarity
This bit enables the user to select the polarity of the clock output on the CK pin in Synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window
1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE=0).

**Note:** If Synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 9 **CPHA**: Clock phase
This bit is used to select the phase of the clock output on the CK pin in Synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see Figure 680 and Figure 681)

0: The first clock transition is the first data capture edge
1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE=0).

**Note:** If Synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 8 **LBCL**: Last bit clock pulse
This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in Synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin
1: The clock pulse of the last data bit is output to the CK pin

**Caution:** The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE=0).

**Note:** If Synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable
Break interrupt mask (break detection using break delimiter).
0: Interrupt is inhibited
1: An interrupt is generated whenever LBDF=1 in the USART_ISR register

**Note:** If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.
Bit 5 **LBDL**: LIN break detection length
- This bit is for selection between 11 bit or 10 bit break detection.
  0: 10-bit break detection
  1: 11-bit break detection
- This bit can only be written when the USART is disabled (UE=0).

**Note**: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 4 **ADDM7**: 7-bit Address Detection/4-bit Address Detection
- This bit is for selection between 4-bit address detection or 7-bit address detection.
  0: 4-bit address detection
  1: 7-bit address detection (in 8-bit data mode)
- This bit can only be written when the USART is disabled (UE=0)

**Note**: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**: When the DIS_NSS bit is set, the NSS pin input is ignored.
- 0: SPI slave selection depends on NSS input pin.
- 1: SPI slave is always selected and NSS input pin is ignored.

**Note**: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous Slave mode enable
- When the SLVEN bit is set, the Synchronous slave mode is enabled.
  0: Slave mode disabled.
  1: Slave mode enabled.

**Note**: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

**50.8.4 USART control register 3 (USART_CR3)**

Address offset: 0x08
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXFTCFG[2:0]</td>
<td>RXFTCFG[2:0]</td>
<td>TCBG</td>
<td>TIE</td>
<td>TXFIE</td>
<td>WUFIE</td>
<td>WUS1</td>
<td>WUS0</td>
<td>SCARCNT[2:0]</td>
<td>DEP</td>
<td>DEM</td>
<td>DDRE</td>
<td>OVR</td>
<td>DIS</td>
<td>ONE</td>
<td>CTSE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DEP</td>
<td>DEM</td>
<td>DDRE</td>
<td>OVR</td>
<td>DIS</td>
<td>ONE</td>
<td>CTSE</td>
<td>CTSE</td>
<td>RTSE</td>
<td>DMAT</td>
<td>DMAR</td>
<td>SCEN</td>
<td>NACK</td>
<td>HDSEL</td>
<td>IRLP</td>
<td>IREN</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Note**: The CPOL, CPHA and LBCL bits must not be written while the transmitter is enabled.
Bits 31:29 **TXFTCFG[2:0]**: TXFIFO threshold configuration

- 000: TXFIFO reaches 1/8 of its depth
- 001: TXFIFO reaches 1/4 of its depth
- 010: TXFIFO reaches 1/2 of its depth
- 011: TXFIFO reaches 3/4 of its depth
- 100: TXFIFO reaches 7/8 of its depth
- 101: TXFIFO becomes empty
- Remaining combinations: Reserved

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]**: Receive FIFO threshold configuration

- 000: Receive FIFO reaches 1/8 of its depth
- 001: Receive FIFO reaches 1/4 of its depth
- 010: Receive FIFO reaches 1/2 of its depth
- 011: Receive FIFO reaches 3/4 of its depth
- 100: Receive FIFO reaches 7/8 of its depth
- 101: Receive FIFO becomes full
- Remaining combinations: Reserved

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated whenever TCBGT=1 in the USART_ISR register

**Note**: If the USART does not support the Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated whenever WUF=1 in the USART_ISR register

**Note**: WUFIE must be set before entering in low-power mode.

**If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.**
Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
- 00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
- 01: Reserved.
- 10: WUF active on start bit detection
- 11: WUF active on RXNE/RXFNE.

This bitfield can only be written when the USART is disabled (UE=0).

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.*

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in Smartcard mode.

In Transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In Reception mode, it specifies the number of erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE=0).

When the USART is enabled (UE=1), this bitfield may only be written to 0x0, in order to stop retransmission.

- 0x0: retransmission disabled - No automatic retransmission in Transmission mode.
- 0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

*Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.*

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

- 0: DE signal is active high.
- 1: DE signal is active low.

This bit can only be written when the USART is disabled (UE=0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.*

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

- 0: DE function is disabled.
- 1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE=0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 50.4: USART implementation on page 2229.*

Bit 13 **DDRE**: DMA Disable on Reception Error

- 0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for Smartcard mode)
- 1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE/RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE=0).

*Note: The reception errors are: parity error, framing error or noise error.*
Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.
This bit can only be written when the USART is disabled (UE=0).

*Note*: This control bit enables checking the communication flow w/o reading the data

Bit 11 **ONEBIT**: One sample bit method enable
This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.
0: Three sample bit method
1: One sample bit method
This bit can only be written when the USART is disabled (UE=0).

Bit 10 **CTSE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF=1 in the USART_ISR register

*Note*: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0).
If the CTS input is asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
This bit can only be written when the USART is disabled (UE=0)

*Note*: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.
This bit can only be written when the USART is disabled (UE=0).

*Note*: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception
Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling Smartcard mode.

- 0: Smartcard mode disabled
- 1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE=0).

**Note:** If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 4 **NACK**: Smartcard NACK enable

- 0: NACK transmission in case of parity error is disabled
- 1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE=0).

**Note:** If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 3 **HDSEL**: Half-duplex selection

Selection of Single-wire Half-duplex mode

- 0: Half-duplex mode is not selected
- 1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE=0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

- 0: Normal mode
- 1: Low-power mode

This bit can only be written when the USART is disabled (UE=0).

**Note:** If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

- 0: IrDA disabled
- 1: IrDA enabled

This bit can only be written when the USART is disabled (UE=0).

**Note:** If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE=1 or ORE=1 or NE=1 or UDR = 1 in the USART_ISR register).

- 0: Interrupt inhibited
- 1: interrupt generated when FE=1 or ORE=1 or NE=1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.
50.8.5  **USART baud rate register (USART_BRR)**

This register can only be written when the USART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BRR[15:0]

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

Bits 15:0 **BRR[15:0]:** USART baud rate

**BRR[15:4]**


**BRR[3:0]**

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

50.8.6  **USART guard time and prescaler register (USART_GTPR)**

Address offset: 0x10

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

**GT[7:0]**

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

**PSC[7:0]**

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
50.8.7 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:8 | GT[7:0]: Guard time value |
| This bitfield is used to program the Guard time value in terms of number of baud clock periods. |
| This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value. |
| This bitfield can only be written when the USART is disabled (UE=0). |
| **Note:** If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. |
| Refer to Section 50.4: USART implementation on page 2229. |

| Bits 7:0 | PSC[7:0]: Prescaler value |
| **Condition:** IrDA low-power and normal IrDA mode |
| PSC[7:0] = IrDA Normal and Low-power baud rate |
| This bitfield is used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency: |
| The source clock is divided by the value given in the register (8 significant bits): |
| 00000000: Reserved - do not program this value |
| 00000001: divides the source clock by 1 |
| 00000010: divides the source clock by 2 |
| ... |
| **Condition:** Smartcard mode |
| PSC[4:0]: Prescaler value |
| This bitfield is used for programming the prescaler for dividing the USART source clock to provide the Smartcard clock. |
| The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency: |
| 00000: Reserved - do not program this value |
| 00001: divides the source clock by 2 |
| 00010: divides the source clock by 4 |
| 00011: divides the source clock by 6 |
| ... |
| This bitfield can only be written when the USART is disabled (UE=0). |
| **Note:** Bits [7:5] must be kept cleared if Smartcard mode is used. |
| This bitfield is reserved and forced by hardware to 0 when the Smartcard and IrDA modes are not supported. Refer to Section 50.4: USART implementation on page 2229. |
Bits 31:24 BLEN[7:0]: Block Length
This bitfield gives the Block length in Smartcard T=1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.
Examples:
BLEN = 0 -> 0 information characters + LEC
BLEN = 1 -> 0 information characters + CRC
BLEN = 255 -> 254 information characters + CRC (total 256 characters))
In Smartcard mode, the Block length counter is reset when TXE=0 (TXFE = 0 in case FIFO mode is enabled).
This bitfield can be used also in other modes. In this case, the Block length counter is reset when RE=0 (receiver disabled) and/or when the EOBCF bit is written to 1.
Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 RTO[23:0]: Receiver timeout value
This bitfield gives the Receiver timeout value in terms of number of bit duration.
In Standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.
In Smartcard mode, this value is used to implement the CWT and BWT. See Smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.
Note: This value must only be programmed once per received character.

Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.
This register is reserved and forced by hardware to "0x00000000" when the Receiver timeout feature is not supported. Refer to Section 50.4: USART implementation on page 2229.

50.8.8 USART request register (USART_RQR)
Address offset: 0x18
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>
Bits 31:5 Reserved, must be kept at reset value.

Bit 4 TXFRQ: Transmit data flush request
When FIFO mode is disabled, writing 1 to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in Smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and Smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 RXFRQ: Receive data flush request
Writing 1 to this bit empties the entire receive FIFO i.e. clears the bit RXFNE. This enables to discard the received data without reading them, and avoid an overrun condition.

Bit 2 MMCRQ: Mute mode request
Writing 1 to this bit puts the USART in Mute mode and resets the RWU flag.

Bit 1 SBKREQ: Send break request
Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKREQ bit.

Bit 0 ABFRREQ: Auto baud rate request
Writing 1 to this bit resets the ABRF and ABRE flags in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

50.8.9 USART interrupt and status register (USART_ISR)
Address offset: 0x1C
Reset value: 0x0XX0 00C0
XX = 28 if FIFO/Smartcard mode enabled
XX = 08 if FIFO enabled and Smartcard mode disabled

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXF</td>
<td>RXF</td>
<td>TCB</td>
<td>RXF</td>
<td>TXF</td>
<td>REA</td>
<td>TEA</td>
<td>WUF</td>
<td>RWU</td>
<td>SBKF</td>
<td>CMF</td>
<td>BUSY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ABR</td>
<td>ABR</td>
<td>UDR</td>
<td>EOB</td>
<td>RTF</td>
<td>CTS</td>
<td>CTSI</td>
<td>LDB</td>
<td>TXF</td>
<td>TC</td>
<td>RXF</td>
<td>IDLE</td>
<td>ORE</td>
<td>NE</td>
<td>FE</td>
<td>PE</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
Bits 31:28  Reserved, must be kept at reset value.

Bit 27  **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG of USART_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit =1 (bit 31) in the USART_CR3 register.

0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26  **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in RXFTCFG in USART_CR3 register is reached. This means that there are (RXFTCFG - 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit =1 (bit 27) in the USART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

**Note:** When the RXFTCFG threshold is configured to 101, RXFT flag is set if 16 data are available i.e. 15 data in the RXFIFO and 1 data in the USART_RDR. Consequently, the 17th received data does not cause an overrun error. The overrun error occurs after receiving the 18th data.

Bit 25  **TCBGT**: Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if the smart card did not send back any NACK. An interrupt is generated if TCBGTIE=1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)
1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

**Note:** If the USART does not support the Smartcard mode, this bit is reserved and kept at reset value. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 50.4: USART implementation on page 2229.

Bit 24  **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.

An interrupt is generated if the RXFFIE bit =1 in the USART_CR1 register.

0: RXFIFO not full.
1: RXFIFO Full.

Bit 23  **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.

An interrupt is generated if the TXFEIE bit =1 (bit 30) in the USART_CR1 register.

0: TXFIFO not empty.
1: TXFIFO empty.
Bit 22  **REACK**: Receive enable acknowledge flag
This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.
It can be used to verify that the USART is ready for reception before entering low-power mode.

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation on page 2229.*

Bit 21  **TEACK**: Transmit enable acknowledge flag
This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.
It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the USART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20  **WUF**: Wake-up from low-power mode flag
This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register. An interrupt is generated if WUFIE=1 in the USART_CR3 register.

*Note: When UESM is cleared, WUF flag is also cleared.*
If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 19  **RWU**: Receiver wake-up from Mute mode
This bit indicates if the USART is in Mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.
When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.
0: Receiver in Active mode
1: Receiver in Mute mode

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation on page 2229.*

Bit 18  **SBKF**: Send break flag
This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.
0: No break character transmitted
1: Break character transmitted

Bit 17  **CMF**: Character match flag
This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register. An interrupt is generated if CMIE=1 in the USART_CR1 register.
0: No Character match detected
1: Character match detected

Bit 16  **BUSY**: Busy flag
This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).
0: USART is idle (no reception)
1: Reception on going
Bit 15  **ABRF**: Auto baud rate flag
This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXFNE and FE are also set in this case)
It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.
*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 14  **ABRE**: Auto baud rate error
This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)
It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.
*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 13  **UDR**: SPI slave underrun error flag
In Slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.
0: No underrun error
1: underrun error
*Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation on page 2229.*

Bit 12  **EOBF**: End of block flag
This bit is set by hardware when a complete block has been received (for example T=1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.
An interrupt is generated if EOBIE = 1 in the USART_CR1 register.
It is cleared by software, writing 1 to EOBCF in the USART_ICR register.
0: End of Block not reached
1: End of Block (number of characters) reached
*Note: If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation on page 2229.*

Bit 11  **RTOF**: Receiver timeout
This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.
An interrupt is generated if RTOIE=1 in the USART_CR2 register.
In Smartcard mode, the timeout corresponds to the CWT or BWT timings.
0: Timeout value not reached
1: Timeout value reached without any data reception
*Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.
The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.
If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.*
Bit 10  **CTS**: CTS flag
This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.
0: CTS line set
1: CTS line reset
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9  **CTSIF**: CTS interrupt flag
This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.
An interrupt is generated if CTSIE=1 in the USART_CR3 register.
0: No change occurred on the CTS status line
1: A change occurred on the CTS status line
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8  **LBDF**: LIN break detection flag
This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.
An interrupt is generated if LBDIE = 1 in the USART_CR2 register.
0: LIN Break not detected
1: LIN break detected
*Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value.*
Refer to Section 50.4: USART implementation on page 2229.

Bit 7  **TXFNF**: TXFIFO not full
TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.
An interrupt is generated if the TXFNFIE bit =1 in the USART_CR1 register.
0: Transmit FIFO is full
1: Transmit FIFO is not full
*Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE is set at the same time). This bit is used during single buffer transmission.*

Bit 6  **TC**: Transmission complete
This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register. The TC flag behaves as follows:
- When TDN = 0, the TC flag is set when the transmission of a frame containing data is complete and when TXE/TXFE is set.
- When TDN is equal to the number of data in the TXFIFO, the TC flag is set when TXFIFO is empty and TDN is reached.
- When TDN is greater than the number of data in the TXFIFO, TC remains cleared until the TXFIFO is filled again to reach the programmed number of data to be transferred.
- When TDN is less than the number of data in the TXFIFO, TC is set when TDN is reached even if the TXFIFO is not empty.
An interrupt is generated if TCIE=1 in the USART_CR1 register.
TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.
Bit 5  **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXFNEIE=1 in the USART_CR1 register.

0: Data is not received
1: Received data is ready to be read.

Bit 4  **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected
1: Idle line is detected

*Note: The IDLE bit is not set again until the RXFNE bit has been set (i.e. a new idle line occurs).*

If Mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3  **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXFNEIE=1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

0: No overrun error
1: Overrun error is detected

*Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.*

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.
Bit 2  **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected
1: Noise is detected

*Note:* This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 50.5.9: Tolerance of the USART receiver to clock deviation on page 2248).

This error is associated with the character in the USART_RDR.

Bit 1  **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected
1: Framing error or break character is detected

*Note:* This error is associated with the character in the USART_RDR.

Bit 0  **PE**: Parity error

This bit is set by hardware when a parity error occurs in Reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error
1: Parity error

*Note:* This error is associated with the character in the USART_RDR.

### 50.8.10 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

#### FIFO mode disabled

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABRF</td>
<td>A BRE</td>
<td>UDR</td>
<td>EOBF</td>
<td>RTOF</td>
<td>CTS</td>
<td>CTSI F</td>
<td>LBD</td>
<td>TXE</td>
<td>TC</td>
<td>RXNE</td>
<td>IDLE</td>
<td>ORE</td>
<td>NE</td>
<td>FE</td>
<td>PE</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

---

RM0481 Universal synchronous/asynchronous receiver transmitter (USART/UART)
Bits 31:26  Reserved, must be kept at reset value.

Bit 25  TCBGT: Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTE=1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)
1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the Smartcard mode, this bit is reserved and kept at reset value. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 50.4: USART implementation on page 2229.

Bits 24:23  Reserved, must be kept at reset value.

Bit 22  REACK: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 21  TEACK: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the USART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20  WUF: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register. An interrupt is generated if WUFIE=1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bit 19  RWU: Receiver wake-up from Mute mode

This bit indicates if the USART is in Mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in Active mode
1: Receiver in Mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation on page 2229.
Bit 18 **SBKF**: Send break flag  
This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.  
0: No break character transmitted  
1: Break character transmitted

Bit 17 **CMF**: Character match flag  
This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.  
An interrupt is generated if CMIE=1 in the USART_CR1 register.  
0: No Character match detected  
1: Character match detected

Bit 16 **BUSY**: Busy flag  
This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).  
0: USART is idle (no reception)  
1: Reception on going

Bit 15 **ABRF**: Auto baud rate flag  
This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXNE and FE are also set in this case).  
It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.  
*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 14 **ABRE**: Auto baud rate error  
This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed).  
It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.  
*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 13 **UDR**: SPI slave underrun error flag  
In Slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.  
0: No underrun error  
1: underrun error  
*Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation on page 2229.*

Bit 12 **EOBF**: End of block flag  
This bit is set by hardware when a complete block has been received (for example T=1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.  
An interrupt is generated if EOBIE = 1 in the USART_CR1 register.  
It is cleared by software, writing 1 to EOBCF in the USART_ICR register.  
0: End of Block not reached  
1: End of Block (number of characters) reached  
*Note: If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation on page 2229.*
Bit 11 **RTOF**: Receiver timeout
This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.
An interrupt is generated if RTOIE=1 in the USART_CR2 register.
In Smartcard mode, the timeout corresponds to the CWT or BWT timings.
0: Timeout value not reached
1: Timeout value reached without any data reception

*Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set. The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.*

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag
This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.
0: CTS line set
1: CTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSIF**: CTS interrupt flag
This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.
An interrupt is generated if CTSIE=1 in the USART_CR3 register.
0: No change occurred on the CTS status line
1: A change occurred on the CTS status line

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 **LBDF**: LIN break detection flag
This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.
An interrupt is generated if LBDIE = 1 in the USART_CR2 register.
0: LIN Break not detected
1: LIN break detected

*Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation on page 2229.*

Bit 7 **TXE**: Transmit data register empty
TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in Smartcard T=0 mode, in case of transmission failure).
An interrupt is generated if the TXEIE bit =1 in the USART_CR1 register.
0: Data register full
1: Data register full
Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data is complete and when TXE is set.

An interrupt is generated if TCIE=1 in the USART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE=1 in the USART_CR1 register.

0: Data is not received
1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected
1: Idle line is detected

Note: *The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line occurs).*

If Mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXNE=1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXNEIE=1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

1: Overrun error is detected

Note: *When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.*

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.
### USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>WUCF</td>
<td>Wake-up from low-power mode clear flag</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

**Bit 20** WUCF: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART_ISR register.

**Note:** If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.

Bits 19:18 Reserved, must be kept at reset value.

**Bit 2** NE: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected
1: Noise is detected

**Note:** This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 50.5.9: Tolerance of the USART receiver to clock deviation on page 2248). This error is associated with the character in the USART_RDR.

**Bit 1** FE: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register. When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame). An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected
1: Framing error or break character is detected

**Note:** This error is associated with the character in the USART_RDR.

**Bit 0** PE: Parity error

This bit is set by hardware when a parity error occurs in Reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register. An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error
1: Parity error

**Note:** This error is associated with the character in the USART_RDR.
Bit 17 **CMCF**: Character match clear flag
Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRF**: SPI slave underrun clear flag
Writing 1 to this bit clears the UDRF flag in the USART_ISR register.
*Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229*

Bit 12 **EOBCF**: End of block clear flag
Writing 1 to this bit clears the EOBF flag in the USART_ISR register.
*Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.*

Bit 11 **RTOCF**: Receiver timeout clear flag
Writing 1 to this bit clears the RTOF flag in the USART_ISR register.
*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.*

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.
*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.*

Bit 8 **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the USART_ISR register.
*Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2229.*

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1 **FE CF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.
50.8.12  USART receive data register (USART_RDR)

Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:9  Reserved, must be kept at reset value.

Bits 8:0  RDR[8:0]: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see Figure 674).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

50.8.13  USART transmit data register (USART_TDR)

Address offset: 0x28
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:9  Reserved, must be kept at reset value.

Bits 8:0  TDR[8:0]: Transmit data value
Contains the data character to be transmitted.
The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 674).
When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF=1.
50.8.14  **USART prescaler register (USART_PRESC)**

This register can only be written when the USART is disabled (UE=0).

Address offset: 0x2C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**W** **R** **R** **R**  **PRESCALER[3:0]**

**Bits 31:4**  Reserved, must be kept at reset value.

**Bits 3:0**  **PRESCALER[3:0]:** Clock prescaler

The USART input clock can be divided by a prescaler factor:

- 0000: input clock not divided
- 0001: input clock divided by 2
- 0010: input clock divided by 4
- 0011: input clock divided by 6
- 0100: input clock divided by 8
- 0101: input clock divided by 10
- 0110: input clock divided by 12
- 0111: input clock divided by 16
- 1000: input clock divided by 32
- 1001: input clock divided by 64
- 1010: input clock divided by 128
- 1011: input clock divided by 256

Remaining combinations: Reserved

**Note:**  When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011 i.e. input clock divided by 256.
## 50.8.15 USART register map

### Table 540. USART register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Field name</th>
<th>Field description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>USART_CR1</td>
<td>FIFOEN</td>
<td>FIFO enable</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M1</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EOBIE</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTOIE</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEAT(4:0)</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FIFOD</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EOB</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTO</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x00</td>
<td>USART_CR1</td>
<td>FIFOEN</td>
<td>FIFO enable</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M1</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EOBIE</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTOIE</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEAT(4:0)</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FIFOD</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EOB</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTO</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x04</td>
<td>USART_CR2</td>
<td>ALEN(7:0)</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALEN(7:0)</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALEN(7:0)</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x08</td>
<td>USART_CR3</td>
<td>TXFTCG</td>
<td>TXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TXFTCG</td>
<td>TXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TXFTCG</td>
<td>TXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x0C</td>
<td>USART_BRR</td>
<td>TXFT</td>
<td>TXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TXFT</td>
<td>TXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TXFT</td>
<td>TXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x10</td>
<td>USART_GTPR</td>
<td>GT(7:0)</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GT(7:0)</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GT(7:0)</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x14</td>
<td>USART_RTOR</td>
<td>BLEN(7:0)</td>
<td>BLEN(7:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTO(23:0)</td>
<td>RTO(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTO(23:0)</td>
<td>RTO(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x18</td>
<td>USART_RQR</td>
<td>TXF</td>
<td>TXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TXF</td>
<td>TXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TXF</td>
<td>TXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x1C</td>
<td>USART_ISR</td>
<td>RXF</td>
<td>RXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RXF</td>
<td>RXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RXF</td>
<td>RXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x1C</td>
<td>USART_ISR</td>
<td>RXF</td>
<td>RXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RXF</td>
<td>RXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RXF</td>
<td>RXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x20</td>
<td>USART_ICR</td>
<td>TXF</td>
<td>TXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TXF</td>
<td>TXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TXF</td>
<td>TXF16/32(23:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x24</td>
<td>USART_RDR</td>
<td>RDR</td>
<td>RDR(8:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RDR</td>
<td>RDR(8:0)</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RDR</td>
<td>RDR(8:0)</td>
<td>0x00000000</td>
</tr>
</tbody>
</table>
Table 540. USART register map and reset values (continued)

| Offset | Register name | Name       | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------|------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x28   | USART_TDR    | TDR[8:0]   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   |            |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x2C   | USART_PRESC  |            |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   |            |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

Refer to Section 2.3 on page 111 for the register boundary addresses.
51 Low-power universal asynchronous receiver transmitter (LPUART)

This section describes the low-power universal asynchronous receiver transmitter (LPUART).

51.1 Introduction

The LPUART is an UART which enables bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to enable UART communications up to 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports Half-duplex Single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

51.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 bauds to 9600 bauds using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data
  Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
• Transfer detection flags:
  – Receive buffer full
  – Transmit buffer empty
  – Busy and end of transmission flags
• Parity control:
  – Transmits parity bit
  – Checks parity of received data byte
• Four error detection flags:
  – Overrun error
  – Noise detection
  – Frame error
  – Parity error
• Interrupt sources with flags
• Multiprocessor communications: wake-up from Mute mode by idle line detection or address mark detection
• Wake-up from Stop mode

51.3 LPUART implementation

The table below describe LPUART implementation. It also includes USARTs and UARTs for comparison.

<table>
<thead>
<tr>
<th>USART modes/features</th>
<th>STM32H563/H573</th>
</tr>
</thead>
<tbody>
<tr>
<td>USART1</td>
<td>FULL</td>
</tr>
<tr>
<td>USART2</td>
<td>FULL</td>
</tr>
<tr>
<td>USART3</td>
<td>FULL</td>
</tr>
<tr>
<td>USART6</td>
<td>FULL</td>
</tr>
<tr>
<td>USART10</td>
<td>FULL</td>
</tr>
<tr>
<td>USART11</td>
<td>FULL</td>
</tr>
<tr>
<td>UART4</td>
<td>BASIC</td>
</tr>
<tr>
<td>UART5</td>
<td>BASIC</td>
</tr>
<tr>
<td>UART7</td>
<td>BASIC</td>
</tr>
<tr>
<td>UART8</td>
<td>BASIC</td>
</tr>
<tr>
<td>UART12</td>
<td>BASIC</td>
</tr>
<tr>
<td>LPUART1</td>
<td>LP</td>
</tr>
<tr>
<td><strong>USART modes/features(^{(1)})</strong></td>
<td>Full feature set</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Hardware flow control for modem</td>
<td>X</td>
</tr>
<tr>
<td>Continuous communication using DMA</td>
<td>X</td>
</tr>
<tr>
<td>Multiprocessor communication</td>
<td>X</td>
</tr>
<tr>
<td>Synchronous mode (Master/Slave)</td>
<td>X</td>
</tr>
<tr>
<td>Smartcard mode</td>
<td>X</td>
</tr>
<tr>
<td>Single-wire Half-duplex communication</td>
<td>X</td>
</tr>
<tr>
<td>IrDA SIR ENDEC block</td>
<td>X</td>
</tr>
<tr>
<td>LIN mode</td>
<td>X</td>
</tr>
<tr>
<td>Dual clock domain</td>
<td>X</td>
</tr>
<tr>
<td>Receiver timeout interrupt</td>
<td>X</td>
</tr>
<tr>
<td>Modbus communication</td>
<td>X</td>
</tr>
<tr>
<td>Auto baud rate detection</td>
<td>X</td>
</tr>
<tr>
<td>Driver Enable</td>
<td>X</td>
</tr>
<tr>
<td><strong>USART data length</strong></td>
<td></td>
</tr>
<tr>
<td>Tx/Rx FIFO</td>
<td>X</td>
</tr>
<tr>
<td>Tx/Rx FIFO size</td>
<td></td>
</tr>
<tr>
<td>Wake-up from low-power mode</td>
<td>(X(^{(2)}))</td>
</tr>
</tbody>
</table>

1. X = supported.
2. Wake-up supported from Stop mode.
51.4 LPUART functional description

51.4.1 LPUART block diagram

Figure 701. LPUART block diagram
51.4.2 LPUART pins and internal signals

Description LPUART input/output pins

- LPUART bidirectional communications
  LPUART bidirectional communications requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):
  - **RX** (Receive Data Input):
    RX is the serial data input.
  - **TX** (Transmit Data Output)
    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In Single-wire mode, this I/O is used to transmit and receive the data.

- RS232 hardware flow control mode
  The following pins are required in RS232 Hardware flow control mode:
  - **CTS** (Clear To Send)
    When driven high, this signal blocks the data transmission at the end of the current transfer.
  - **RTS** (Request to send)
    When it is low, this signal indicates that the USART is ready to receive data.

- RS485 hardware flow control mode
  The **DE** (Driver Enable) pin is required in RS485 Hardware control mode. This signal activates the transmission mode of the external transceiver.

Refer to Table 543 and Table 544 for the list of LPUART input/output pins and internal signals.

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPUART_RX</td>
<td>Input</td>
<td>Serial data receive input</td>
</tr>
<tr>
<td>LPUART_TX</td>
<td>Output</td>
<td>Transmit data output</td>
</tr>
<tr>
<td>LPUART_CTS</td>
<td>Input</td>
<td>Clear to send</td>
</tr>
<tr>
<td>LPUART_RTS</td>
<td>Output</td>
<td>Request to send</td>
</tr>
<tr>
<td>LPUART_DE(1)</td>
<td>Output</td>
<td>Driver enable</td>
</tr>
</tbody>
</table>

1. LPUART_DE and LPUART_RTS share the same pin.

Description LPUART input/output signals

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usart_pclk</td>
<td>Input</td>
<td>APB clock</td>
</tr>
<tr>
<td>lpuart_ker_ck</td>
<td>Input</td>
<td>LPUART kernel clock</td>
</tr>
<tr>
<td>lpuart_wkup</td>
<td>Output</td>
<td>LPUART provides a wake-up interrupt</td>
</tr>
</tbody>
</table>
51.4.3 LPUART clocks

The simplified block diagram given in Figure 701 shows two fully independent clock domains:

- **The lpuart_pclk clock domain**
  
  The `lpuart_pclk` clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.

- **The lpuart_ker_ck kernel clock domain**

  The `lpuart_ker_ck` is the LPUART clock source. It is independent of the `lpuart_pclk` and delivered by the RCC. So, the LPUART registers can be written/read even when the `lpuart_ker_ck` is stopped.

  When the dual clock domain feature is not supported, the `lpuart_ker_ck` is the same as the `lpuart_pclk` clock.

  There is no constraint between `lpuart_pclk` and `lpuart_ker_ck`: `lpuart_ker_ck` can be faster or slower than `lpuart_pclk`, with no more limitation than the ability for the software to manage the communication fast enough.

51.4.4 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see Figure 675).

- 7-bit character length: M[1:0] = ‘10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

An **Idle character** is interpreted as an entire frame of “1”s. (The number of “1”’s includes the number of stop bits).

A **Break character** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.
Figure 702. LPUART word length programming

9-bit word length (M = 01), 1 Stop bit

Data frame

Possible Parity bit

Start bit Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Bit8 Stop bit

Idle frame

Break frame

8-bit word length (M = 00), 1 Stop bit

Data frame

Possible Parity bit

Start bit Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Stop bit

Idle frame

Break frame

7-bit word length (M = 10), 1 Stop bit

Data frame

Possible Parity bit

Start bit Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Stop bit

Idle frame

Break frame

** LBCL bit controls last data clock pulse
51.4.5 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bits wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

*Note:* The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in LPUART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bitfields.

  In this case, the RXFT flag is set in the LPUART_ISR register. This means that RXFTCFG data have been received: 1 data in LPUART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to ‘101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received: FIFO size - 1 data in the RXFIFO and 1 data in the LPUART_RDR. As a result, the next received data does not set the overrun flag.

- The Tx interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bitfields.

51.4.6 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

**Character transmission**

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Figure 701*).

When FIFO mode is enabled, the data written to the LPUART_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.
Note: The TE bit must be set before writing the data to be transmitted to the LPUART_TDR. The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.
An idle frame is sent after the TE bit is enabled.

Configurable stop bits

The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13, 12).

- **1 stop bit**: This is the default value of number of stop bits.
- **2 Stop bits**: This is supported by normal LPUART, Single-wire and Modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

**Figure 703. Configurable stop bits**

8-bit Word length (M[1:0]=00 bit is reset)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>BIT1</td>
<td>BIT2</td>
<td>BIT3</td>
<td>BIT4</td>
<td>BIT5</td>
</tr>
<tr>
<td>STOP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**a) 1 Stop bit**

- **CLOCK**
- Possible parity bit
- Next start bit
- Next data frame
- **LBCL bit controls last data clock pulse**

**b) 2 Stop bits**

- Possible parity bit
- Next start bit
- Next data frame

Character transmission procedure

To transmit a character, follow the sequence below:
1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAT) in LPUART_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in Section 51.4.13: Continuous communication using DMA and LPUART.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the LPUART_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
   - When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   - When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO. Write operations to the LPUART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.

8. When the last data is written to the LPUART_TDR register, wait until TC=1. This indicates that the transmission of the last frame is complete.
   - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
   - When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode disabled:
  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:
  - the data have been moved from the LPUART_TDR register to the shift register and data transmission has started;
  - the LPUART_TDR register is empty;
  - the next data can be written to the LPUART_TDR register without overwriting the previous data.

  The TXE flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the LPUART_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO Not Full) flag is set by hardware to indicate that:
  - the TXFIFO is not full;
  - the LPUART_TDR register is empty;
  - the next data can be written to the LPUART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the
LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE is case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC=1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see Figure 704: TC/TXE behavior when transmitting).

Figure 704. TC/TXE behavior when transmitting

Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see Figure 702).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.
When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

**Idle characters**

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

### 51.4.7 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART.CR1 register.

**Start bit detection**

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still 0. If the start sample is at 1, then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

**Character reception**

During an LPUART reception, data are shifted in least significant bit first (default configuration) through the RX pin. In this mode, the LPUART.RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

**Character reception procedure**

To receive a character, follow the sequence below:

1. Program the M bits in LPUART.CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART.BRR
3. Program the number of stop bits in LPUART.CR2.
4. Enable the LPUART by writing the UE bit in LPUART.CR1 register to 1.
5. Select DMA enable (DMAR) in LPUART.CR3 if multibuffer communication is to take place. Configure the DMA register as explained in [Section 51.4.13: Continuous communication using DMA and LPUART](#).
6. Set the RE bit LPUART.CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART.RDR returns the oldest data entered in the RXFIFO.
When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.

- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
- In Multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the Receive Data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty i.e. there is a data in the RXFIFO to be read.
- In Single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

**Break character**

When a break character is received, the USART handles it as a framing error.

**Idle character**

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.
Overrun error

- FIFO mode disabled
  An overrun error occurs when a character is received when RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.
  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:
  - the ORE bit is set;
  - the RDR content is not lost. The previous data is available when a read to LPUART_RDR is performed;
  - the shift register is overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXNEIE bit or EIE bit is set.

- FIFO mode enabled
  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.
  Data can not be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.
  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:
  - the ORE bit is set;
  - the first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
  - the shift register is overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T
When the FIFO mode is disabled, there are two possibilities
- if RXNE=1, then the last valid data is stored in the receive register (RDR) and can be read,
- if RXNE=0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.

Selecting the clock source

The choice of the clock source is done through the Clock Control system (see Section Reset and clock controller (RCC)). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.
When the dual clock domain and the wake-up from low-power mode features are supported, the lpuart_ker_ck clock source can be configured in the RCC (see Section Reset and clock controller (RCC)). Otherwise, the lpuart_ker_ck is the same as lpuart_pclk.

The lpuart_ker_ck can be divided by a programmable factor in the LPUART_PRES register.

**Figure 705. lpuart_ker_ck clock divider block diagram**

Some lpuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and Wake-up mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

*Note: There is no noise detection for data.*

**Framing error**

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:
- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the LPUART_RDR register.
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.
Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2; it can be either 1 or 2 in Normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample i.e. during the second stop bit. The first stop bit is not checked for framing error.

51.4.8 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register.

\[
Tx/Rx \text{ baud} = \frac{256 \times \text{lpuart_ker_ck_pres}}{\text{LPUARTDIV}}
\]

LPUARTDIV is defined in the LPUART_BRR register.

**Note:** The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence the baud rate register value must not be changed during communication.

It is forbidden to write values lower than 0x300 in the LPUART_BRR register. f\(_{\text{CK}}\) must range from 3 x baud rate to 4096 x baud rate.

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock. For example, if the LPUART clock source frequency is 100 MHz, the maximum baud rate that can be reached is about 33 Mbauds.

<table>
<thead>
<tr>
<th>S.No</th>
<th>Desired</th>
<th>Actual</th>
<th>Value programmed in the baud rate register</th>
<th>% Error = (Calculated - Desired) B.rate / Desired B.rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3 kbaud</td>
<td>0.3 kbaud</td>
<td>0x6D3A</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.6 kbaud</td>
<td>0.6 kbaud</td>
<td>0x369D</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1200 bauds</td>
<td>1200.087 bauds</td>
<td>0x1B4E</td>
<td>0.007</td>
</tr>
<tr>
<td>4</td>
<td>2400 bauds</td>
<td>2400.17 bauds</td>
<td>0xDA7</td>
<td>0.007</td>
</tr>
<tr>
<td>5</td>
<td>4800 bauds</td>
<td>4801.72 bauds</td>
<td>0x6D3</td>
<td>0.035</td>
</tr>
<tr>
<td>6</td>
<td>9600 kbauds</td>
<td>9608.94 bauds</td>
<td>0x369</td>
<td>0.093</td>
</tr>
</tbody>
</table>

Table 545. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz
### Table 546. Error calculation for programmed baud rates at $f_{CK} = 100$ MHz

<table>
<thead>
<tr>
<th>S.No</th>
<th>Desired</th>
<th>Actual</th>
<th>Value programmed in the baud rate register</th>
<th>$%$ Error = (Calculated - Desired) $B$.rate / Desired $B$.rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>38400 bauds</td>
<td>38400,04 bauds</td>
<td>A2C2A</td>
<td>0,0001</td>
</tr>
<tr>
<td>2</td>
<td>57600 bauds</td>
<td>57600,06 bauds</td>
<td>6C81C</td>
<td>0,0001</td>
</tr>
<tr>
<td>3</td>
<td>115200 bauds</td>
<td>115200,12 bauds</td>
<td>3640E</td>
<td>0,0001</td>
</tr>
<tr>
<td>4</td>
<td>230400 bauds</td>
<td>230400,23 bauds</td>
<td>1B207</td>
<td>0,0001</td>
</tr>
<tr>
<td>5</td>
<td>460800 bauds</td>
<td>460804,61 bauds</td>
<td>D903</td>
<td>0,001</td>
</tr>
<tr>
<td>6</td>
<td>921600 bauds</td>
<td>921625,81 bauds</td>
<td>6C81</td>
<td>0,0028</td>
</tr>
<tr>
<td>7</td>
<td>4000 kbauds</td>
<td>4000000,00 bauds</td>
<td>1900</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>10000 kbauds</td>
<td>10000000,00 bauds</td>
<td>A00</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>20000 kbauds</td>
<td>20000000,00 bauds</td>
<td>500</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>30000 kbauds</td>
<td>33032258,06 bauds</td>
<td>307</td>
<td>0,1</td>
</tr>
</tbody>
</table>
51.4.9 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes which contribute to the total deviation are:

- **DTRA**: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- **DQUANT**: error due to the baud rate quantization of the receiver
- **DREC**: deviation of the receiver local oscillator
- **DTCL**: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

\[
DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}
\]

where

- **DWU** is the error due to sampling point deviation when the wake-up from low-power mode is used.

When \(M[1:0] = 01\):

\[
DWU = \frac{t_{WULPUART}}{11 \times \text{Tbit}}
\]

When \(M[1:0] = 00\):

\[
DWU = \frac{t_{WULPUART}}{10 \times \text{Tbit}}
\]

When \(M[1:0] = 10\):

\[
DWU = \frac{t_{WULPUART}}{9 \times \text{Tbit}}
\]

\(t_{WULPUART}\) is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in Table 547:

- Number of Stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

### Table 547. Tolerance of the LPUART receiver

<table>
<thead>
<tr>
<th>M bits</th>
<th>768 &lt; BRR &lt; 1024</th>
<th>1024 &lt; BRR &lt; 2048</th>
<th>2048 &lt; BRR &lt; 4096</th>
<th>4096 ≤ BRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits (M=00), 1 Stop bit</td>
<td>1.82%</td>
<td>2.56%</td>
<td>3.90%</td>
<td>4.42%</td>
</tr>
<tr>
<td>9 bits (M=01), 1 Stop bit</td>
<td>1.69%</td>
<td>2.33%</td>
<td>2.53%</td>
<td>4.14%</td>
</tr>
<tr>
<td>7 bits (M=10), 1 Stop bit</td>
<td>2.08%</td>
<td>2.86%</td>
<td>4.35%</td>
<td>4.42%</td>
</tr>
<tr>
<td>8 bits (M=00), 2 Stop bit</td>
<td>2.08%</td>
<td>2.86%</td>
<td>4.35%</td>
<td>4.42%</td>
</tr>
<tr>
<td>9 bits (M=01), 2 Stop bit</td>
<td>1.82%</td>
<td>2.56%</td>
<td>3.90%</td>
<td>4.42%</td>
</tr>
<tr>
<td>7 bits (M=10), 2 Stop bit</td>
<td>2.34%</td>
<td>3.23%</td>
<td>4.92%</td>
<td>4.42%</td>
</tr>
</tbody>
</table>
Note: The data specified in Table 547 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M=’01 or 9-bit times when M = ‘10).

51.4.10 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all non addressed receivers.

The non addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart_ker_ck cycles), otherwise Mute mode might remain active.

When the Mute mode is enabled:
- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in LPUART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:
- Idle Line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The LPUART enters Mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of Mute mode behavior using Idle line detection is given in Figure 706.
Figure 706. Mute mode using Idle line detection

Note: If the MMRQ is set while the IDLE character has already elapsed, Mute mode is not entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a 1 otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters Mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters Mute mode.

The LPUART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The LPUART exits from Mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in Mute mode.

An example of Mute mode behavior using address mark detection is given in Figure 707.
51.4.11 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in Table 548.

### Table 548: LPUART frame formats

<table>
<thead>
<tr>
<th>M bits</th>
<th>PCE bit</th>
<th>LPUART frame(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

### Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

### Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).
Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

51.4.12 LPUART Single-Wire Half-Duplex Communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register.

The LPUART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to 1:
- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

51.4.13 Continuous Communication Using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 50.4: USART implementation on page 2229 to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 50.5.7. To perform continuous communication. When FIFO is disabled, clear the TXE/RXNE flags in the LPUART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to Section Direct memory access controller) to the LPUART_TDR register whenever the TXFNF flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.

2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.

3. Configure the total number of bytes to be transferred to the DMA control register.

4. Configure the channel priority in the DMA register.

5. Configure DMA interrupt generation after half/ full transfer as required by the application.

6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.

7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In Transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the LPUART communication is complete. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

![Figure 708. Transmission using DMA](image-url)
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to section Direct memory access controller (DMA)) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register.
5. Configure interrupt generation after half/full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 709. Reception using DMA

Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in Multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt.
enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

51.4.14 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The Figure 710 shows how to connect 2 devices in this mode:

Figure 710. Hardware flow control between 2 LPUARTs

RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE=1), then RTS is deasserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. Figure 711 shows an example of communication with RTS flow control enabled.

Figure 711. RS232 RTS flow control

Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.
RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. Figure 712 shows an example of communication with CTS flow control enabled.

![Figure 712. RS232 CTS flow control](image)

Note: For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.
The LPUART DEAT and DEDT are expressed in LPUART clock source (f\(_{\text{CK}}\)) cycles:

- The Driver enable assertion time equals
  - \((1 + (\text{DEAT} \times P)) \times f_{\text{CK}}, \text{if } P \neq 0\)
  - \((1 + \text{DEAT}) \times f_{\text{CK}}, \text{if } P = 0\)
- The Driver enable de-assertion time equals
  - \((1 + (\text{DEDT} \times P)) \times f_{\text{CK}}, \text{if } P \neq 0\)
  - \((1 + \text{DEDT}) \times f_{\text{CK}}, \text{if } P = 0\)

where \(P = \text{BRR}[20:11]\)

### 51.4.15 LPUART low-power management

The LPUART has advanced low-power mode functions that enable it to transfer properly data even when the lpuart_pclk clock is disabled. The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set. When the usart_pclk is gated, the LPUART provides a wake-up interrupt (usart_wkup) if a specific action requiring the activation of the usart_pclk clock is needed:

- If FIFO mode is disabled
  - lpuart_pclk clock has to be activated to empty the LPUART data register.
  - In this case, the lpuart_wkup interrupt source is the RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled
  - lpuart_pclk clock has to be activated
    - to fill the TXFIFO
    - or to empty the RXFIFO
  - In this case, the lpuart_wkup interrupt source can be:
    - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
    - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
    - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.
  - This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.
  - To avoid overrun/underrun errors and transmit/receive data in low-power mode, the lpuart_wkup interrupt source can be one of the following events:
    - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
    - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.
  - For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time to receive a single byte across the line.
  - Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.
Alternatively, a specific `lpuart_wkup` interrupt may be selected through the WUS bitfields. When the wake-up event is detected, the WUF flag is set by hardware and `lpuart_wkup` interrupt is generated if the WUFIE bit is set.

**Note:**

Before entering low-power mode, make sure that no LPUART transfer is ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or in an active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure the LPUART is actually enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled upon exit from low-power mode.

When FIFO is enabled, the wake-up from low-power mode on address match is only possible when Mute mode is enabled.

**Using Mute mode with low-power mode**

If the LPUART is put into Mute mode before entering low-power mode:

- Wake-up from Mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from Mute mode on address match is used, then the low-power mode wake-up source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in Mute mode upon address match and wake up from low-power mode.

**Note:**

When FIFO management is enabled, Mute mode is used with wake-up from low-power mode without any constraints (i.e. the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

**Wake-up from low-power mode when LPUART kernel clock lpuart_ker_ck is OFF in low-power mode**

If during low-power mode, the lpuart_ker_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the lpuart_ker_ck clock to be switched ON thanks to the lpuart_ker_ck_req signal. The lpuart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, the usart_ker_ck is switched OFF again, the MCU is not waken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of wake-up event programmed to “address match detection” and FIFO management disabled.

_Figure 713_ shows the behavior when the wake-up event is verified.
Figure 713. Wake-up event verified (wake-up event = address match, FIFO disabled)

- Address match event WUF = '1'
- LPUART sends a wakeup event to the MCU
- Data reception goes on

Figure 714 shows the behavior when the wake-up event is not verified.

Figure 714. Wake-up event not verified (wake-up event = address match, FIFO disabled)

- Address does not match

Note: The above figures are valid when address match or any received frame is used as wake-up event. In the case the wake-up event is the start bit detection, the LPUART sends the wake-up event to the MCU at the end of the start bit.
Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see Section 51.4.9: Tolerance of the LPUART receiver to clock deviation).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000. In these conditions, according to Table 547: Tolerance of the LPUART receiver, the LPUART receiver tolerance equals 3.41%.

\[
D_{WU\text{max}} = \frac{t_{\text{WULPUART}}}{(11 \times T_{\text{bit Min}})}
\]

\[
T_{\text{bit Min}} = \frac{t_{\text{WULPUART}}}{(11 \times D_{WU\text{max}})}
\]

where \(t_{\text{WULPUART}}\) is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

\[
t_{\text{WULPUART}} = 3 \mu\text{s} \text{ (values provided only as examples; for correct values, refer to the device datasheet).}
\]

\[
D_{WU\text{max}} = 3.41\% - 1\% = 2.41\%
\]

\[
T_{\text{bit min}} = \frac{3 \mu\text{s}}{(11 \times 2.41\%)} = 11.32 \mu\text{s}.
\]

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: \(1/11.32 \mu\text{s} = 88.36 \text{kbauds}.

### 51.5 LPUART in low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. LPUART interrupts cause the device to exit Sleep mode.</td>
</tr>
<tr>
<td>Stop(^1)</td>
<td>The content of the LPUART registers is kept. The LPUART is able to wake up the microcontroller from Stop mode when the LPUART is clocked by an oscillator available in Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The LPUART peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

\(^1\) Refer to Section 51.3: LPUART implementation to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.
## 51.6 LPUART interrupts

Refer to *Table 550* for a detailed description of all LPUART interrupt requests.

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop(1) modes</th>
<th>Exit from Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit data register empty</td>
<td>TXE</td>
<td>TXEIE</td>
<td>Write TDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmit FIFO Not Full</td>
<td>TXFNIE</td>
<td>TXFIFO full</td>
<td></td>
<td></td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmit FIFO Empty</td>
<td>TXFE</td>
<td>TXFEIE</td>
<td>Write TDR or write 1 in TXFRQ</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmit FIFO threshold reached</td>
<td>TXFT</td>
<td>TXFTE</td>
<td>Write TDR</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTS interrupt</td>
<td>CTSIF</td>
<td>CTSIE</td>
<td>Write 1 in CTSCF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmission Complete</td>
<td>TC</td>
<td>TCIE</td>
<td>Write TDR or write 1 in TCCF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive data register not empty (data ready to be read)</td>
<td>RXNE</td>
<td>RXNEIE</td>
<td>Read RDR or write 1 in RXFRQ</td>
<td>Yes</td>
<td></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Receive FIFO Not Empty</td>
<td>RXFNE</td>
<td>RXFNEIE</td>
<td>Read RDR until RXFIFO empty or write 1 in RXFRQ</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive FIFO Full</td>
<td>RXFF(2)</td>
<td>RXFIE</td>
<td>Read RDR</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive FIFO threshold reached</td>
<td>RXFT</td>
<td>RXFTE</td>
<td>Read RDR</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overrun error detected</td>
<td>ORE</td>
<td>RXNIEE/RXFNEIE</td>
<td>Write 1 in ORECF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idle line detected</td>
<td>IDLE</td>
<td>IDLEIE</td>
<td>Write 1 in IDLECF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parity error</td>
<td>PE</td>
<td>PEIE</td>
<td>Write 1 in PECF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noise error in multibuffer communication.</td>
<td>NE</td>
<td></td>
<td>Write 1 in NFCF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overrun error in multibuffer communication.</td>
<td>ORE(3)</td>
<td>EIE</td>
<td>Write 1 in ORECF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framing Error in multibuffer communication.</td>
<td>FE</td>
<td></td>
<td>Write 1 in FECF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Character match</td>
<td>CMF</td>
<td>CMIE</td>
<td>Write 1 in CMCF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wake-up from low-power mode</td>
<td>WUF</td>
<td>WUFIE</td>
<td>Write 1 in WUC</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the Wake-up from Stop mode feature. Refer to Section 51.3: LPUART implementation for the list of supported Stop modes.

2. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).

3. When OVRDIS = 0.

51.7 LPUART registers

Refer to Section 1.2 on page 101 for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

51.7.1 LPUART control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

**FIFO mode enabled**

```
+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+
<table>
<thead>
<tr>
<th>RXFIE</th>
<th>TXFEIE</th>
<th>FIFO EN</th>
<th>M1</th>
<th>Res.</th>
<th>Res.</th>
<th>DEAT[4:0]</th>
<th>DEDT[4:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+
| 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  |
| Res. | CMIE| MME | M0  | WAKE| PCE | PS  | PEIE | TXFN| FIE| TCIE | RXFN| EIE| IDLEIE| TE  | RE  | UESM| UE  |
|      | rw | rw | rw  | rw  | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+
```

**Bit 31 RXFIE:** RXFIFO Full interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated when RXFF=1 in the LPUART_ISR register

**Bit 30 TXFEIE:** TXFIFO empty interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated when TXFE=1 in the LPUART_ISR register

**Bit 29 FIFOEN:** FIFO mode enable

- This bit is set and cleared by software.
- 0: FIFO mode is disabled.
- 1: FIFO mode is enabled.
Bit 28  **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

- M[1:0] = 00: 1 Start bit, 8 Data bits, n Stop bit
- M[1:0] = 01: 1 Start bit, 9 Data bits, n Stop bit
- M[1:0] = '10: 1 Start bit, 7 Data bits, n Stop bit

This bit can only be written when the LPUART is disabled (UE=0).

*Note:* In 7-bit data length mode, the Smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26  Reserved, must be kept at reset value.

Bits 25:21  **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer to Section 50.5.21: RS232 Hardware flow control and RS485 Driver Enable. This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 20:16  **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in lpuart_ker_ck clock cycles. For more details, refer to Section 51.4.14: RS232 Hardware flow control and RS485 Driver Enable. If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed. This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15  Reserved, must be kept at reset value.

Bit 14  **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13  **MME**: Mute mode enable

This bit activates the Mute mode function of the LPUART. When set, the LPUART can switch between the active and Mute modes, as defined by the WAKE bit. It is set and cleared by software.

- 0: Receiver in Active mode permanently
- 1: Receiver can switch between Mute mode and Active mode.

Bit 12  **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description). This bit can only be written when the LPUART is disabled (UE=0).

Bit 11  **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from Mute mode. It is set or cleared by software.

- 0: Idle line
- 1: Address mark

This bitfield can only be written when the LPUART is disabled (UE=0).
Bit 10  **PCE**: Parity control enable
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if \(M=1\); 8th bit if \(M=0\)) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).
- 0: Parity control disabled
- 1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 9  **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
- 0: Even parity
- 1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 8  **PEIE**: PE interrupt enable
This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever PE=1 in the LPUART_ISR register

Bit 7  **TXFNFIE**: TXFIFO not full interrupt enable
This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated whenever TXFNF =1 in the LPUART_ISR register

Bit 6  **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever TC=1 in the LPUART_ISR register

Bit 5  **RXFNEIE**: RXFIFO not empty interrupt enable
This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated whenever ORE=1 or RXFNE=1 in the LPUART_ISR register

Bit 4  **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever IDLE=1 in the LPUART_ISR register

Bit 3  **TE**: Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
- 0: Transmitter is disabled
- 1: Transmitter is enabled

**Note:** During transmission, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.
Bit 2 \textbf{RE:} Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 \textbf{UESM:} LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.
When this bit is set, the LPUART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: USART not able to wake up the MCU from low-power mode.
1: USART able to wake up the MCU from low-power mode.

\textit{Note:} It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 \textbf{UE:} LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.
0: LPUART prescaler and outputs disabled, low-power mode
1: LPUART enabled

\textit{Note:} To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

### 51.7.2 LPUART control register 1 [alternate] (LPUART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

\textbf{FIFO mode disabled}

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>CMIE</td>
<td>MME</td>
<td>M0</td>
<td>WAKE</td>
<td>PCE</td>
<td>PS</td>
<td>PEIE</td>
<td>TXIEIE</td>
<td>TCIE</td>
<td>RXNEIE</td>
<td>IDLEIE</td>
<td>TE</td>
<td>RE</td>
<td>UESM</td>
<td>UE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable
   This bit is set and cleared by software.
   0: FIFO mode is disabled.
   1: FIFO mode is enabled.

Bit 28 **M1**: Word length
   This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
   \[ M[1:0] = \begin{cases} 00 & \text{1 Start bit, 8 Data bits, n Stop bit} \\ 01 & \text{1 Start bit, 9 Data bits, n Stop bit} \\ 10 & \text{1 Start bit, 7 Data bits, n Stop bit} \end{cases} \]
   This bit can only be written when the LPUART is disabled (UE=0).

**Note:** In 7-bit data length mode, the Smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time
   This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer **Section 50.5.21: RS232 Hardware flow control and RS485 Driver Enable**.
   This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time
   This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in lpuart_ker_ck clock cycles. For more details, refer **Section 51.4.14: RS232 Hardware flow control and RS485 Driver Enable**.
   If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.
   This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable
   This bit is set and cleared by software.
   0: Interrupt is inhibited
   1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable
   This bit activates the Mute mode function of the LPUART. When set, the LPUART can switch between the active and Mute modes, as defined by the WAKE bit. It is set and cleared by software.
   0: Receiver in Active mode permanently
   1: Receiver can switch between Mute mode and Active mode.

Bit 12 **M0**: Word length
   This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).
   This bit can only be written when the LPUART is disabled (UE=0).
Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from Mute mode. It is set or cleared by software.

- 0: Idle line
- 1: Address mark

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software.

- 0: Parity control disabled
- 1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

- 0: Even parity
- 1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever PE=1 in the LPUART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated whenever TXE =1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever TC=1 in the LPUART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated whenever ORE=1 or RXNE=1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated whenever IDLE=1 in the LPUART_ISR register
Bit 3 TE: Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled

Note: During transmission, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to ‘1’. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.
In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 RE: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 UESM: LPUART enable in low-power mode
When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.
When this bit is set, the LPUART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: USART not able to wake up the MCU from low-power mode.
1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 UE: LPUART enable
When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.
0: LPUART prescaler and outputs disabled, low-power mode
1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

51.7.3 LPUART control register 2 (LPUART_CR2)
Address offset: 0x04
Reset value: 0x0000 0000
Bits 31:24 **ADD[7:0]**: Address of the LPUART node

These bits give the address of the LPUART node in Mute mode or a character code to be recognized in low-power or Run mode:

- In Mute mode: they are used in multiprocessor communication to wake up from Mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.

- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.

- In Run mode with Mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set. These bits can only be written when the reception is disabled (RE = 0) or when the USART is disabled (UE = 0).

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

- 0: data is transmitted/received with data bit 0 first, following the start bit.
- 1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 18 **DATAINV**: Binary data inversion

This bit is set and cleared by software.

- 0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
- 1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 17 **TXINV**: TX pin active level inversion

This bit is set and cleared by software.

- 0: TX pin signal works using the standard logic levels (V_DD =1/idle, Gnd=0/mark)
- 1: TX pin signal values are inverted. ((V_DD =0/mark, Gnd=1/idle). This enables the use of an external inverter on the TX line.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 16 **RXINV**: RX pin active level inversion

This bit is set and cleared by software.

- 0: RX pin signal works using the standard logic levels (V_DD =1/idle, Gnd=0/mark)
- 1: RX pin signal values are inverted. ((V_DD =0/mark, Gnd=1/idle). This enables the use of an external inverter on the RX line.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15 **SWAP**: Swap TX/RX pins

This bit is set and cleared by software.

- 0: TX/RX pins are used as defined in standard pinout
- 1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 14 Reserved, must be kept at reset value.
Bits 13:12 STOP[1:0]: STOP bits
  These bits are used for programming the stop bits.
  00: 1 stop bit
  01: Reserved.
  10: 2 stop bits
  11: Reserved
  This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 ADDM7: 7-bit Address Detection/4-bit Address Detection
  This bit is for selection between 4-bit address detection or 7-bit address detection.
  0: 4-bit address detection
  1: 7-bit address detection (in 8-bit data mode)
  This bit can only be written when the LPUART is disabled (UE=0)
  Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address
  (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

51.7.4 LPUART control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>DEP</td>
<td>DEM</td>
<td>DDRE</td>
<td>OVRDI</td>
<td>Res.</td>
<td>CTSIE</td>
<td>CTSE</td>
<td>RTSE</td>
<td>DMAT</td>
<td>DMAR</td>
<td>Res.</td>
<td>Res.</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.
Bits 31:29 **TXFTCFG[2:0]**: TXFIFO threshold configuration
- 000: TXFIFO reaches 1/8 of its depth.
- 001: TXFIFO reaches 1/4 of its depth.
- 110: TXFIFO reaches 1/2 of its depth.
- 100: TXFIFO reaches 7/8 of its depth.
- 101: TXFIFO becomes empty.
- Remaining combinations: Reserved.

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]**: Receive FIFO threshold configuration
- 000: Receive FIFO reaches 1/8 of its depth.
- 001: Receive FIFO reaches 1/4 of its depth.
- 110: Receive FIFO reaches 1/2 of its depth.
- 011: Receive FIFO reaches 3/4 of its depth.
- 100: Receive FIFO reaches 7/8 of its depth.
- 101: Receive FIFO becomes full.
- Remaining combinations: Reserved.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated whenever WUF=1 in the LPUART_ISR register

Note: **WUFIE must be set before entering in low-power mode.**
If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 51.3: LPUART implementation on page 2317.

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (Wake-up from low-power mode flag).
- 00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
- 01: Reserved.
- 10: WUF active on start bit detection
- 11: WUF active on RXNE/RXFNE.

This bitfield can only be written when the LPUART is disabled (UE=0).

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 51.3: LPUART implementation on page 2317.

Bits 19:16 Reserved, must be kept at reset value.
Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 13 **DDRE**: DMA Disable on Reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE=0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE=0).

Note: This control bit enables checking the communication flow w/o reading the data.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF=1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
This bit can only be written when the LPUART is disabled (UE=0)

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

- 1: DMA mode is enabled for reception
- 0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection

Selection of Single-wire Half-duplex mode

- 0: Half-duplex mode is not selected
- 1: Half-duplex mode is selected

This bit can only be written when the LPUART is disabled (UE=0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NE=1 in the LPUART_ISR register).

- 0: Interrupt is inhibited
- 1: An interrupt is generated when FE=1 or ORE=1 or NE=1 in the LPUART_ISR register.

**51.7.5 LPUART baud rate register (LPUART_BRR)**

This register can only be written when the LPUART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **BRR[19:0]**: LPUART baud rate division (LPUARTDIV)

Note: It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

Provided that LPUART_BRR must be ≥ 0x300 and LPUART_BRR is 20 bits, a care must be taken when generating high baud rates using high fck values. fck must be in the range [3 x baud rate..4096 x baud rate].
51.7.6  LPUART request register (LPUART_RQR)

Address offset: 0x18
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 4</td>
<td>TXFRQ: Transmit data flush request</td>
</tr>
<tr>
<td></td>
<td>This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).</td>
</tr>
<tr>
<td>Note:</td>
<td>In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.</td>
</tr>
<tr>
<td>Bit 3</td>
<td>RXFRQ: Receive data flush request</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the RXNE flag.</td>
</tr>
<tr>
<td></td>
<td>This enables discarding the received data without reading it, and avoid an overrun condition.</td>
</tr>
<tr>
<td>Bit 2</td>
<td>MMRQ: Mute mode request</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.</td>
</tr>
<tr>
<td>Bit 1</td>
<td>SBKRQ: Send break request</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.</td>
</tr>
<tr>
<td>Note:</td>
<td>If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.</td>
</tr>
<tr>
<td>Bit 0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

51.7.7  LPUART interrupt and status register (LPUART_ISR)

Address offset: 0x1C
Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 30:24</td>
<td>TXFT</td>
</tr>
<tr>
<td></td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>r</td>
</tr>
</tbody>
</table>
Bits 31:28  Reserved, must be kept at reset value.

Bit 27  **TXFT**: TXFIFO threshold flag  
This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG in LPUART_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFIE bit = 1 (bit 31) in the LPUART_CR3 register.  
0: TXFIFO does not reach the programmed threshold.  
1: TXFIFO reached the programmed threshold.

Bit 26  **RXFT**: RXFIFO threshold flag  
This bit is set by hardware when the RXFIFO reaches the threshold programmed in RXFTCFG in LPUART_CR3 register i.e. the Receive FIFO contains RXFTCFG data. An interrupt is generated if the RXFTIE bit = 1 (bit 27) in the LPUART_CR3 register.  
0: Receive FIFO does not reach the programmed threshold.  
1: Receive FIFO reached the programmed threshold.

Bit 25  Reserved, must be kept at reset value.

Bit 24  **RXFF**: RXFIFO Full  
This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register. An interrupt is generated if the RXFFIE bit = 1 in the LPUART_CR1 register.  
0: RXFIFO is not Full.  
1: RXFIFO is Full.

Bit 23  **TXFE**: TXFIFO Empty  
This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register. An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the LPUART_CR1 register.  
0: TXFIFO is not empty.  
1: TXFIFO is empty.

Bit 22  **REACK**: Receive enable acknowledge flag  
This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.  
It can be used to verify that the LPUART is ready for reception before entering low-power mode.  
**Note**: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21  **TEACK**: Transmit enable acknowledge flag  
This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.  
It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the LPUART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20  **WUF**: Wake-up from low-power mode flag  
This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register. An interrupt is generated if WUFEIE = 1 in the LPUART_CR3 register.  
**Note**: When UESM is cleared, WUF flag is also cleared.  
**If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 51.3: LPUART implementation on page 2317.**
Bit 19 **RWU**: Receiver wake-up from Mute mode
This bit indicates if the LPUART is in Mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.
When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.
0: Receiver in Active mode
1: Receiver in Mute mode
*Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.*

Bit 18 **SBKF**: Send break flag
This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRO bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.
0: No break character transmitted
1: Break character transmitted

Bit 17 **CMF**: Character match flag
This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.
An interrupt is generated if CMIE=1 in the LPUART_CR1 register.
0: No Character match detected
1: Character match detected

Bit 16 **BUSY**: Busy flag
This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).
0: LPUART is idle (no reception)
1: Reception on going

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag
This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.
0: CTS line set
1: CTS line reset
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSIF**: CTS interrupt flag
This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.
An interrupt is generated if CTSIE=1 in the LPUART_CR3 register.
0: No change occurred on the CTS status line
1: A change occurred on the CTS status line
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 Reserved, must be kept at reset value.
Bit 7  **TXFNF**: TXFIFO not full  
TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFIE bit = 1 in the LPUART_CR1 register.

0: Data register is full/Transmit FIFO is full.
1: Data register/Transmit FIFO is not full.

*Note*: This bit is used during single buffer transmission.

Bit 6  **TC**: Transmission complete  
This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register. The TC flag behaves as follows:

- When TDN = 0, the TC flag is set when the transmission of a frame containing data is complete and when TXF is set.
- When TDN is equal to the number of data in the TXFIFO, the TC flag is set when TXFIFO is empty and TDN is reached.
- When TDN is greater than the number of data in the TXFIFO, TC remains cleared until the TXFIFO is filled again to reach the programmed number of data to be transferred.
- When TDN is less than the number of data in the TXFIFO, TC is set when TDN is reached even if the TXFIFO is not empty.

An interrupt is generated if TCIE=1 in the LPUART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the LPUART_ICR register or by writing to the LPUART_TDR register.

Bit 5  **RXFNE**: RXFIFO not empty  
RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO.

It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXFNEIE=1 in the LPUART_CR1 register.

0: Data is not received
1: Received data is ready to be read.

Bit 4  **IDLE**: Idle line detected  
This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected
1: Idle line is detected

*Note*: The IDLE bit is not set again until the RXFNE bit has been set (i.e. a new idle line occurs).

If Mute mode is enabled (MME=1), IDLE is set if the LPUART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.
Bit 3  **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXFNEIE=1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

*Note:* When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2  **NE**: Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected
1: Noise is detected

*Note:* This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

This error is associated with the character in the LPUART_RDR.

Bit 1  **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR3 register.

0: No Framing error is detected
1: Framing error or break character is detected

*Note:* This error is associated with the character in the LPUART_RDR.

Bit 0  **PE**: Parity error

This bit is set by hardware when a parity error occurs in Reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error
1: Parity error

*Note:* This error is associated with the character in the LPUART_RDR.
51.7.8 LPUART interrupt and status register [alternate] (LPUART_ISR)

Address offset: 0x1C
Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

### FIFO mode disabled

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res.</td>
<td>Res.</td>
<td>Res.</td>
<td>Res.</td>
<td>CTS</td>
<td>CTSF</td>
<td>TXE</td>
<td>TC</td>
<td>RXNE</td>
<td>IDLE</td>
<td>ORE</td>
<td>NE</td>
<td>FE</td>
<td>PE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:23 Reserved, must be kept at reset value.

- **Bit 22** REACK: Receive enable acknowledge flag
  - This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.
  - It can be used to verify that the LPUART is ready for reception before entering low-power mode.
  - **Note:** If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

- **Bit 21** TEACK: Transmit enable acknowledge flag
  - This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.
  - It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the LPUART_CR1 register, in order to respect the TE=0 minimum period.

- **Bit 20** WUF: Wake-up from low-power mode flag
  - This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register. An interrupt is generated if WUFIE=1 in the LPUART_CR3 register.
  - **Note:** When UESM is cleared, WUF flag is also cleared.
  - If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 51.3: LPUART implementation on page 2317.

- **Bit 19** RWU: Receiver wake-up from Mute mode
  - This bit indicates if the LPUART is in Mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.
  - When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.
  - **0:** Receiver in Active mode
  - **1:** Receiver in Mute mode
  - **Note:** If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.
Bit 18 **SBKF**: Send break flag
This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.
0: No break character transmitted
1: Break character transmitted

Bit 17 **CMF**: Character match flag
This bit is set by hardware, when the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.
An interrupt is generated if CMIE=1 in the LPUART_CR1 register.
0: No Character match detected
1: Character match detected

Bit 16 **BUSY**: Busy flag
This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).
0: LPUART is idle (no reception)
1: Reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag
This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.
0: CTS line set
1: CTS line reset
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSF**: CTS interrupt flag
This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.
An interrupt is generated if CTSIE=1 in the LPUART_CR3 register.
0: No change occurred on the CTS status line
1: A change occurred on the CTS status line
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE**: Transmit data register empty
TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.
An interrupt is generated if the TXEIE bit =1 in the LPUART_CR1 register.
0: Data register is full/Transmit FIFO is full.
1: Data register/Transmit FIFO is not full.
*Note: This bit is used during single buffer transmission.*

Bit 6 **TC**: Transmission complete
This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data is complete and when TXE is set.
An interrupt is generated if TCIE=1 in the LPUART_CR1 register.
TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.
Bit 5 RXNE: Read data register not empty
RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been transferred to the LPUART_RDR register. It is cleared by a read to the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. An interrupt is generated if RXNEIE=1 in the LPUART_CR1 register.
0: Data is not received
1: Received data is ready to be read.

Bit 4 IDLE: Idle line detected
This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.
0: No Idle line is detected
1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line occurs).

If Mute mode is enabled (MME=1), IDLE is set if the LPUART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 ORE: Overrun error
This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXNE=1 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.
An interrupt is generated if RXNEIE=1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.
1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.
This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.
Bit 2 **NE**: Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected
1: Noise is detected

*Note:* This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame). An interrupt is generated if EIE = 1 in the LPUART_CR3 register.

0: No Framing error is detected
1: Framing error or break character is detected

*Note:* In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in Reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error
1: Parity error

*Note:* In FIFO mode, this error is associated with the character in the LPUART_RDR.

### 51.7.9 LPUART interrupt flag clear register (LPUART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART_ISR register.

*Note:* If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 51.3: LPUART implementation on page 2317.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

---

**Notes:**

The LPUART interrupt flag clear register (LPUART_ICR) is used to clear various interrupt flags in the LPUART module. Each bit in the register corresponds to a specific interrupt or error condition in the LPUART module. Writing a 1 to a bit in this register clears the corresponding flag in the LPUART_ISR register, which may cause an interrupt to occur. The register is located at address offset 0x20 and its reset value is 0x0000 0000.

### Additional Notes:

- **NE (Start bit noise detection flag):** This flag is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software by setting the NFCF bit in the LPUART_ICR register.

- **FE (Framing error):** This flag is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software by setting the FECF bit in the LPUART_ICR register.

- **PE (Parity error):** This flag is set by hardware when a parity error occurs in Reception mode. It is cleared by software by setting the PECF bit in the LPUART_ICR register.

- **WUCF (Wake-up from low-power mode clear flag):** Writing 1 to this bit clears the WUF flag in the USART_ISR register. If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value.

- **CMCF (Character match clear flag):** Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

---

**References:**

- Section 51.3: LPUART implementation on page 2317.
Bit 9 **CTSCF**: CTS clear flag  
Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag  
Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag  
Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag  
Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NECF**: Noise detected clear flag  
Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag  
Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag  
Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

### 51.7.10 LPUART receive data register (LPUART_RDR)

Address offset: 0x24  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:9** Reserved, must be kept at reset value.

**Bits 8:0** **RDR[8:0]**: Receive data value  
Contains the received data character.  
The RDR register provides the parallel interface between the input shift register and the internal bus (see Figure 701).  
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.
51.7.11 LPUART transmit data register (LPUART_TDR)

Address offset: 0x28
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 701).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

*Note: This register must be written only when TXE/TXFNF=1.*

51.7.12 LPUART prescaler register (LPUART_PRESC)

This register can only be written when the LPUART is disabled (UE=0).

Address offset: 0x2C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **PRESCALER[3:0]**

The value in this register is used to divide the baud rate generator clock before being applied to the baud rate generator. This register is only accessible when the LPUART is disabled (UE=0).
Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]:** Clock prescaler

The LPUART input clock can be divided by a prescaler:

- 0000: input clock not divided
- 0001: input clock divided by 2
- 0010: input clock divided by 4
- 0011: input clock divided by 6
- 0100: input clock divided by 8
- 0101: input clock divided by 10
- 0110: input clock divided by 12
- 0111: input clock divided by 16
- 1000: input clock divided by 32
- 1001: input clock divided by 64
- 1010: input clock divided by 128
- 1011: input clock divided by 256

Remaining combinations: Reserved.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011 i.e. input clock divided by 256.

51.7.13 LPUART register map

### Table 551. LPUART register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | LPUART_CR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | FIFO mode enabled | FIFOEN | TXIE | RXIE | DEAT[4:0] | DEWD[4:0] | M1 | M0 | WAKE | PS | PCE | PEIE | TXFNFIE | TCIE | RXFNEIE | IDLEIE | TE | RE | UESM | UE |
| Reset value | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 |
| 0x00   | LPUART_CR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | FIFO mode disabled | FIFOEN | TXIE | RXIE | DEAT[4:0] | DEWD[4:0] | M1 | M0 | WAKE | PS | PCE | PEIE | TXFNFIE | TCIE | RXFNEIE | IDLEIE | TE | RE | UESM | UE |
| Reset value | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 |
| 0x04   | LPUART_CR2    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | ADD[7:0]      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 |
| 0x08   | LPUART_CR3    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | TFXTFC[8:0]  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 |
| 0x0C   | LPUART_BRR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | BRR[19:0]     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 | 0 0 0 0 |
| 0x10- 0x14 | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Offset | Register name | Offset 25 | Offset 24 | Offset 23 | Offset 22 | Offset 21 | Offset 20 | Offset 19 | Offset 18 | Offset 17 | Offset 16 | Offset 15 | Offset 14 | Offset 13 | Offset 12 | Offset 11 | Offset 10 | Offset 9 | Offset 8 | Offset 7 | Offset 6 | Offset 5 | Offset 4 | Offset 3 | Offset 2 | Offset 1 | Offset 0 |
|--------|---------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 0x18   | LPUART_RQR    | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value |
| 0x1C   | LPUART_ISR    | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value |
| 0x1C   | LPUART_ISR    | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value |
| 0x20   | LPUART_ICR    | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value |
| 0x24   | LPUART_RDR    | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value |
| 0x28   | LPUART_TDR    | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value |
| 0x2C   | LPUART_PRES   | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value | reset value |

Refer to Section 2.3 for the register boundary addresses.
52 Serial peripheral interface (SPI)

52.1 Introduction

The serial peripheral interface (SPI) can be used to communicate with external devices while using the specific synchronous protocol. The SPI protocol supports half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master or slave and is capable of operating in multi slave or multi master configurations. The device configured as master provides communication clock (SCK) to the slave device. The Slave select (SS) and ready (RDY) signals can be applied optionally just to setup communication with concrete slave and to assure it handles the data flow properly. The Motorola data format is used by default, but some other specific modes are supported as well.

52.2 SPI main features

- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- From 4- up to 32-bit data size selection or fixed to 8-bit multiples
- Multi master or multi slave mode capability
- Dual clock domain, the peripheral kernel clock is independent from the APB bus clock
- Baud rate prescaler up to kernel frequency/2 or bypass from RCC in Master mode
- Protection of configuration and setting
- Hardware or software management of SS for both master and slave
- Adjustable minimum delays between data and between SS and data flow
- Configurable SS signal polarity and timing, MISO x MOSI swap capability
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Programmable number of data within a transaction to control SS and CRC
- Dedicated transmission and reception flags with interrupt capability
- SPI Motorola and TI formats support
- Hardware CRC feature can verify integrity of the communication at the end of transaction by:
  - Adding CRC value in Tx mode
  - Automatic CRC error checking for Rx mode
- Error detection with interrupt capability in case of data overrun, CRC error, data underrun, the mode fault and frame error, depending upon the operating mode
- Two multiples of 8-bit embedded Rx and Tx FIFOs (FIFO size depends on instance)
- Configurable FIFO thresholds (data packing)
- Capability to handle data streams by system DMA controller
- Configurable behavior for slave underrun condition (support of cascaded circular buffers)
- Optional status pin RDY signalizing the slave device ready to handle the data flow
52.3 **SPI implementation**

*Table 552* describes the SPI implementation. The instances are applied either with a full set or a limited set of features.

<table>
<thead>
<tr>
<th>SPI feature</th>
<th>SPI2S1, SPI2S2, SPI2S3 (full feature set instances)</th>
<th>SPI4, SPI5, SPI6 (full feature set instances)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data and CRC size</td>
<td>Configurable from 4 to 32-bit</td>
<td>Configurable from 4 to 16-bit</td>
</tr>
<tr>
<td>CRC computation</td>
<td>CRC polynomial length configurable from 5 to 33-bit</td>
<td>CRC polynomial length configurable from 5 to 17-bit</td>
</tr>
<tr>
<td>Size of FIFOs</td>
<td>16x8-bit</td>
<td>8x8-bit</td>
</tr>
<tr>
<td>Number of data control (TSIZE)</td>
<td>Up to 65536</td>
<td>Up to 65536</td>
</tr>
<tr>
<td>I2S feature</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Autonomous in Stop modes with wakeup capability</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Autonomous in LP-Stop and Standby modes with wakeup capability</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

*Note:* For detailed information about instances capabilities to exit from Stop and Standby modes refer to *Table 556: SPI wake-up and interrupt requests.*
52.4 SPI functional description

52.4.1 SPI block diagram

The SPI enables a synchronous, serial communication between the MCU and external devices. The application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main SPI elements and their interactions are shown in Figure 715.

Figure 715. SPI/I2S block diagram

The simplified scheme of Figure 715 shows three fully independent clock domains:

- the **spi_pclk** clock domain
- the **spi_ker_ck** kernel clock domain
- the serial interface clock domain

All the control and status signals between these domains are strictly synchronized. There is no specific constraint concerning the frequency ratio between these clock signals. The user has to consider a ratio compatible with the data flow speed to avoid data underrun or overrun events.
The \textbf{spi\_pclk} clock signal feeds the peripheral bus interface. It must be active when accesses to the SPI registers are required.

The SPI working in Slave mode handles a data flow using the serial interface clock derived from the external SCK signal provided by the external master SPI device. This is why the SPI slave is able to receive and send data even when the \textbf{spi\_pclk} and \textbf{spi\_ker\_ck} clock signals are inactive. As a consequence, a specific slave logic working within the serial interface clock domain needs some additional traffic to be setup correctly (for example when underrun or overrun is evaluated, see \textit{Section 52.5.2} for details). This cannot be done when the bus becomes idle. In some cases the slave even requires the clock generator working (see \textit{Section 52.5.1}).

When the SPI works as master, it needs \textbf{spi\_ker\_ck} kernel clock coming from RCC active during communication to feed the serial interface clock via the clock generator where it can be divided by prescaler or bypassed optionally. The signal is then provided to slaves via the SCK pin and internally to the serial interface domain of the master.

\textbf{52.4.2 SPI pins and internal signals}

Up to five I/O pins are dedicated to SPI communication with external devices.

- **MISO:** master in / slave out data. In the general case, this pin is used to transmit data in Slave mode and receive data in Master mode.
- **MOSI:** master out / slave in data. In the general case, this pin is used to transmit data in Master mode and receive data in Slave mode.
- **SCK:** serial clock output pin for SPI masters and input pin for SPI slaves.
- **SS:** slave select pin. Depending on the SPI and SS settings, this pin can be used to either:
  - select an individual slave device for communication
  - synchronize the data frame, or
  - detect a conflict between multiple masters
  See \textit{Section 52.4.7} for details.
- **RDY:** optional status pin signaling slave FIFO occupancies and so the slave availability to continue the transaction without any risk of data flow corruption. It can be checked by master to control temporal suspension of the ongoing communication.

All these pins (except RDY) are shared in the I2S mode. This mode features additional I2S specific MCK signal. For more details about I2S signals see \textit{Section 52.9.2}.

The SPI bus enables the communication between one master device and one or more slave devices. The bus consists of at least two wires: one for the clock signal and the other for synchronous data transfer. Other signals are optional and can be added depending on the data exchange between SPI nodes and their communication control management.

Refer to \textit{Table 553} and \textit{Table 554} for the list of SPI input / output pins and internal signals.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|}
\hline
Pin name & I/O type & Description \\
\hline
MISO/SDI\(^{(2)}\) & Input/output & Master data input / slave data output \\
\hline
MOSI/SDO\(^{(2)}\) & Input/output & Master data output / slave data input \\
\hline
SCK/CK & Input/output & Master clock output / slave clock input \\
\hline
\end{tabular}
\caption{SPI/I2S input/output pins\(^{(1)}\)}
\end{table}
52.4.3 SPI communication general aspects

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use 2 or 3 wires (with software SS management) or 3 or 4 wires (with hardware SS management). The communication is always initiated and controlled by the master. The master provides a clock signal on the SCK line and selects or synchronizes slave(s) for communication by SS line when it is managed by hardware. The data between the master and the slave flow on the MOSI and/or MISO lines.

52.4.4 Communications between one master and one slave

The communication flow may use one of 3 possible modes: the full-duplex (3 wires) mode, half-duplex (2 wires) mode or the simplex (2 wires) mode. The SS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the SS signal can be helpful in this configuration to synchronize the data flow and it is used by default for some specific SPI modes (for example the TI mode).

The next optional RDY signal can help to assure correct management of all the transacted data at slave side.
Full-duplex communication

By default, the SPI is configured for full-duplex communication (bits COMM[1:0] = 00 in the SPI_CFG2 register). In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During the SPI communication, the data are shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line simultaneously. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 716. Full-duplex single master/ single slave application

1. To apply SS pins interconnection is not mandatory to make the SPI interface working (see Section 52.4.7 for details).
2. RDY signal provided by slave can be read by master optionally.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting COMM[1:0] = 11 in the SPI_CFG2 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data are synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the HDDIR bit in their SPI_CR1 registers. Note that the SPI must be disabled when changing direction of the communication. In this configuration, the MISO pin at master and the MOSI pin at slave are free for other application uses and act as GPIOs.
1. To apply SS pins interconnection is not mandatory to make the SPI interface working (see Section 52.4.7 for details).

2. In this configuration, the MISO pin at master and MOSI pin at slave can be used as GPIOs.

3. A critical situation can happen when the communication direction is not changed synchronously between two nodes working in bidirectional mode. The new transmitter accesses the common data line while the former transmitter still keeps an opposite value on the line (the value depends on the SPI configuration and communicated data). The nodes can conflict temporarily with opposite output levels on the line until the former transmitter changes its data direction setting. It is suggested to insert a serial resistance between MISO and MOSI pins in this mode to protect the conflicting outputs and limit the current flow between them.

4. RDY signal provided by slave can be read by master optionally.

**Simplex communications**

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the COMM[1:0] field in the SPI_CFG2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO or MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode:** COMM[1:0] = 01
  The master in transmit-only mode generates the clock as long as there are data available in the TxFIFO and the master transfer is ongoing.
  The slave in transmit-only mode sends data as long as it receives a clock on the SCK pin and the SS pin (or software managed internal signal) is active (see Section 52.4.7).

- **Receive-only mode:** COMM[1:0] = 10
  In Master mode, the MOSI output is disabled and may be used as GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by software explicitly requesting this by setting the CSUSP bit in the SPI_CR1 register or automatically when the RxFIFO is full, when the MASRX bit in the SPI_CR1 is set.
  In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see Section 52.4.7).

**Note:** In whatever Master and Slave modes, the data pin dedicated for transmission can be replaced by the data pin dedicated for reception and vice versa by changing the IOSWP bit value in the SPI_CFG2 register (This bit may only be modified when the SPI is disabled).

Any simplex communication can be replaced by a variant of the half duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled, while the HDDIR bit is never changed) or by full duplex control when unused data line and corresponding data flow is ignored.
Figure 718. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode)

1. SS pins interconnection is not mandatory to make the SPI interface working (see Section 52.4.7).
2. In this configuration, both the MISO pins can be used as GPIOs.
3. RDY signal provided by slave can be read by master optionally.

52.4.5 Standard multislave communication

In a configuration with two or more independent slaves, the master uses a star topology with dedicated GPIO pins to manage the chip select lines for each slave separately (see Figure 719.). The master must select one of the slaves individually by pulling low the GPIO connected to the slave SS input (only one slave can control data on common MISO line at a given time). When this is done, a communication between the master and the selected slave is established. Except the simplicity, the advantage of this topology is that a specific SPI configuration can be applied for each slave as all the communication sessions are performed separately just within single master-slave pair. Optionally, when there is no need to read any information from slaves, the master can transmit the same information to the multiple slaves.
1. Master single SS pin hardware output functionality cannot support this topology (to be replaced by set of GPIOs under SW control) and user should avoid SPI AF setting at the pin (see Section 52.4.7 for details).

2. If the application cannot assure that no more than a single SS active signal is provided by the master at a given time, it is better to configure MISO pins into open-drain configuration with an external pull up on the MISO line to prevent conflicts between interconnected outputs of the slaves. Else, a push-pull configuration can be applied without extra resistor (see I/O alternate function input/output (GPIO) section).

3. RDY signals can be read by master from the slaves optionally.

The master can handle the SPI communication with all the slaves in time when a circular topology is applied (see Figure 720). All the slaves behave like simple shift registers applied in serial chain under control of common slave select (SS) and clock (SCK) signals. All the information is shifted simultaneously around the circle while returning back to the master. Sessions have fixed the length where the number of data frames transacted by the master is equal to the number of slaves.

Then when a first data frame is transacted in the chain, the master just sends the information dedicated for the last slave node in the chain, via the first slave node input, while the first information received by the master comes from the last node output at this time. Correspondingly, the lastly transacted data finishing the session is dedicated for the first slave node while its firstly outgoing data just reaches the master input, after its circling
around the chain passing through all the other slaves during the session.
The data format configuration and clock setting must be the same for all the nodes in the
chain in this topology. As the receive and transmit shift registers are separated internally, a
trick with intentional underrun must be applied to the TxFIFO slaves when the information is
transacted between the receiver and the transmitter by hardware.
In this case, the transmission underrun feature is configured in a mode repeating lastly
received data frame (UDRCFG=1). A session can start optionally with a single data pattern
written into the TxFIFO by each slave (usually slave status information is applied) before the
session starts. In this case the underrun happens in fact after this first data frame is
transacted. To be able to clear the internal underrun condition immediately and restart the
session by the TxFIFO content again, the user must disable and enable the SPI between
the sessions and must fill the TxFIFO by a new single data pattern (to overcome the
propagating delay of the clearing raised in case the underrun is cleared in a standard way
by the UDRC bit).

Figure 720. Master and three slaves connected in circular (daisy chain) topology

1. The underrun feature is used by the slaves in this configuration when the slaves are able to transmit data
   received previously into the Rx shift register once their TxFIFOs become empty.
2. RDY signals can be read by master optionally either separately or configured as open drain outputs (while
   RDIOP=0) and connected together with a pull up resistor as a common chain ready status overdriven by
   the slowest device.
52.4.6 Multimaster communication

Unless the SPI bus is not designed primarily for a multimaster capability, it is possible to use a built-in feature that detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, the SS pin is used configured in hardware input mode. The connection of more than two SPI nodes working in this mode is impossible, as only one node can apply its output on a common data line at a given time.

When the nodes are not active, both stay in Slave mode by default. When a node wants to take control on the bus, it switches itself into Master mode and applies active level on the slave select input of the other node via the dedicated GPIO pin. After the session is completed, the active slave select signal is released and the node mastering the bus temporary returns back to passive Slave mode waiting for a next session to start.

If both nodes raise their mastering request at the same time, a bus conflict event appears (see mode fault MODF event). The user can apply some simple arbitration process (for example postpone next attempt by predefined different time-outs applied in both nodes).

Figure 721. Multimaster application

1. The SS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as passive node is configured as a slave.

2. The RDY signal is not used in this communication.

52.4.7 Slave select (SS) pin management

In Slave mode, the SS works as a standard ‘chip select’ input and lets the slave communicate with the master. In Master mode, the SS can be used either as an output or an input. As an input it can prevent a multi master bus collision, and as an output it can drive a slave select signal of a single slave. The SS signal can be managed internally (software management of the SS input) or externally when both the SS input and output are associated with the SS pin (hardware SS management). The user can configure which level of this input/output external signal (present on the SS pin) is considered as active one by the SSIOP bit setting. SS level is considered as active if it is equal to SSIOP.
The hardware or software slave select management can be set using the SSM bit in the SPI_CFG2 register:

- **Software SS management (SSM = 1):** in this configuration, slave select information is driven internally by the SSI bit value in the register SPI_CR1. The external SS pin is free for other application uses (as GPIO or other alternate function).

- **Hardware SS management (SSM = 0):** in this case, there are two possible configurations. The configuration used depends on the SS output configuration (SSOE bit in register SPI_CFG2).
  - **SS output enable (SSOE = 1):** this configuration is only used when the MCU is set as master. The SS pin is managed by the hardware. The functionality is tied to CSTART and EOT control. As a consequence, the master must apply proper TSIZE>0 setting to control the SS output correctly. Even if SPI AF is not applied at the SS pin (it can be used as a standard GPIO then), keep anyway SSOE = 1 to assure default SS input level and prevent any mode fault evaluation at input of the master SS internal logic applicable at a multimaster topology exclusively.
    a) When SSOM = 0 and SP = 000, the SS signal is driven to the active level as soon as the master transfer starts (CSTART = 1) and it is kept active until its EOT flag is set or the transmission is suspended.
    b) When SP = 001, a pulse is generated as defined by the TI mode.
    c) When SSOM = 1, SP = 000 and MIDI > 1 the SS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (1 to 14).
    d) SS input is forced to non active state internally at master to prevent its any mode fault.
  - **SS output disable (SSM = 0, SSOE = 0):**
    a) If the micro-controller is acting as the master on the bus, this configuration allows multi master capability. If the SS pin is pulled into an active level in this mode, the SPI enters Master mode fault state and the SPI device is automatically reconfigured in Slave mode (MASTER = 0).
    b) In Slave mode, the SS pin works as a standard ‘chip select’ input and the slave is selected while the SS line is at its active level.

**Note:** The purpose of automatic switching into Slave mode at mode fault condition is to avoid the possible conflicts on data and clock line. As the SPE is automatically reset in this condition, both Rx and Tx FIFOs are flushed and current data is lost.

When the SPI slave is enabled in the hardware SS management mode, all the traffic is ignored even in case of the SS is found at active level. They are ignored until the slave detects a start of the SS signal (transition from non-active to active level) just synchronizing the slave with the master. This is because the hardware management mode cannot be used when the external SS pin is fixed. There is no such protection in the SS software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.
When the hardware output SS control is applied (SSM = 0, SSOE = 1), by configuration of the MIDI[3:0] and MSSI[3:0] bitfields, the user can control the timing of the SS signal between data frames and can insert an extra delay at the beginning of every transaction (to separate the SS and clock starts). This can be useful when the slave needs to slow down the flow to obtain sufficient room for correct data handling (see Figure 723).

**Figure 723. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0)**

2. CPHA = 0, CPOL = 0, SSIOP = 0, LSBFRST = 0.

Additionally, bit SSOM = 1 setting invokes specific mode which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] must be set greater then one SPI period). Some configuration examples are shown in Figure 724.
Figure 724. SS interleaving pulses between data (S$OE = 1$, SS$OM = 1$, SSM = 0)
2. SS interleaves between data when MIDI[3:0] > 1 wide of the interleaving pulse is always one SCK period less then gap provided between frames (defined by MIDI parameter). If MIDI is set the frames are separated by single SCK period but no interleaving pulse appears on SS.

52.4.8 Ready pin (RDY) management

The status of the slave capability to handle data, can be checked on the RDY pin. By default, a low level indicates that the slave is not ready for transaction. The reason can be that slave's TxFIFO is empty, RxFIFO full or the SPI is disabled. An active level of the signal can be selected by the RDIOP bit. If the master continues or starts to communicate with the slave when it indicates a not ready status, the transaction fails great probably.

The logic to control the RDY output is rather complex, tied closely with TSIZE and DSIZE settings. The RDY reaction is more pessimistic and sensitive to TxFIFO becoming nearly empty and/or RxFIFO nearly full during a frame transaction. This pessimistic logic is suppressed at the end of a transaction only when RDY stays active, despite TxFIFO becomes fully empty and/or RxFIFO becomes fully occupied. The target is to prevent any data corruption and inform master in time that it is necessary to suspend the transaction temporarily till the next transacted data can be processed safely again. When RDY signal input is enabled at master side, master suspends the communication once the slave indicates not ready status. This prevents the master to complete transaction of an ongoing frame which just empties slave's TxFIFO or full fills its RxFIFO till a next data is written and/or read there (despite the frame still can be completed without any constraint). It can make a problem if TSIZE = 0 configuration is applied at slave because slave then never evaluates end of transaction (which suppresses the not ready status just when the last data is sent). Then the user has to release the RxFIFO and/or write additional (even dummy) data to TxFIFO by software at slave side to release the not RDY signal, unblock ST master and so enable it to continue at the communication suspended at middle of a frame occasionally.

When RDY is not used by the master, it must be disabled (RDIOM = 0). Then an internal logic of the master simulates the slave status always ready. In this case, the RDIOP bit setting has no meaning.

Due to synchronization between clock domains and evaluation of the RDY logic on both master and slave sides, the RDY pin feature is not reliable and cannot be used when the size of data frames is configured shorter than 8-bit.

52.4.9 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slave devices must follow the same communication format and be synchronized correctly.

Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPI_CFG2 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data are being transferred. This bit affects both Master and Slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.
If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edges (dotted lines in Figure 725).

Figure 725, shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit. The idle state of SCK must correspond to the polarity selected in the SPI_CFG2 register (by pulling the SCK pin up if CPOL = 1 or pulling it down if CPOL = 0).

Figure 725. Data clock timing diagram

1. The order of data bits depends on LSBFRST bit setting.
Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFIRST bit in SPI_CFG2 register.

At instance with full feature set, the data frame size is chosen by using the DSIZE[4:0] bits at SPI_CFG1 register. It can be set from 4-bit up to 32-bit length and the setting applies for both transmission and reception. When the SPI_TXDR/SPI_RXDR registers are accessed, data frames are always right-aligned into either a byte (if the data fit into a byte), a half-word or a word (see Figure 726).

If the access is a multiple of the configured data size, data packing is applied automatically. During communication, only bits within the data frame are clocked and transferred.

Figure 726. Data alignment when data size is not equal to 8-, 16- or 32-bit

<table>
<thead>
<tr>
<th>DSIZE &lt;= 8-bits</th>
<th>9-bits &lt;= DSIZE &lt;= 16-bits</th>
<th>17-bits &lt;= DSIZE &lt;= 32-bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>data is right aligned on byte Example: DSIZE[4:0]=00011</td>
<td>data is right aligned on half-word Example: DSIZE[4:0]=01101</td>
<td>data is right aligned on word Example: DSIZE[4:0]=11010</td>
</tr>
</tbody>
</table>

Note: The minimum data length is 4 bits. If a data length of less than 4 bits is selected, it is forced to an 4-bit data frame size.

At the instance with limited set of features, data size is fixed to multiply 8-bit up to maximum data length (depends on instance) in according to DSIZE[4:3] bits value. If the SPI_TXDR or SPI_RXDR are accessed by wider access (a multiply of the configured the data size), data packing is applied automatically.

52.4.10 Configuring the SPI

The configuration procedure is almost the same for the master and the slave. For specific mode setups, follow the dedicated chapters. When a standard communication must be initialized, perform these steps:

1. Write the proper GPIO registers: configure GPIO alternate functions at MOSI, MISO, SCK, SS and RDY pins if applied.
2. Write into the SPI_CFG1 and SPI_CFG2 registers and set up proper values of all 'not reserved' bits and bitfields, prior SPI is enabled, with the following exceptions:
   a) The SSOM, MASRX, SSOE, RDIOM, MBR[2:0], BPASS, MIDI[3:0], MSSI[3:0] bits are taken into account in Master mode only, the MSSI[3:0] bits take effect when the SSOE bit is set, the RDIOP bit takes no effect when the RDIOM bit is not set in Master mode. When slave is configured at TI mode, MBR[2:0] setting is considered, too.
   b) UDRCFG is taken into account in Slave mode only.
   c) CRCSIZE[4:0] is required if CRCEN is set.
d) CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI and MIDI are not required in TI mode.
e) Once the AFCNTR bit is set in the SPI_CFG2 register, all the SPI outputs start to be propagated onto the associated GPIO pins regardless the peripheral enable. So any later configuration changes of the SPI_CFG1 and SPI_CFG2 registers can affect the level of signals on these pins.

3. Write to the SPI_CR2 register to select the length of the transfer, if it is not known TSIZE must be programmed to zero.
4. Write to SPI_CRCPOLY and into TCRCINI, RCRCINI and CRC33_17 bits at the SPI_CR1 register to configure the CRC polynomial and CRC calculation if needed.
5. Configure DMA streams dedicated for the SPI Tx and Rx in DMA registers if the DMA streams are used (see chapter Communication using DMA).
6. Configure SSI, HDDIR and MASRX at SPI_CR1 register if required.
7. Program the IOLOCK bit in the SPI_CFG1 register if the configuration protection is required (for safety).

52.4.11 Enabling the SPI

It is recommended to configure and enable the SPI slave before the master sends the clock. But there is no impact if the configuration and enabling procedure is done while a traffic is ongoing on the bus, assuming that the SS signal is managed by hardware at slave or kept inactive by slave’s software when the software management of the SS signal is applied (see Section 52.4.7). The data register of the slave transmitter should contain data to be sent before the master starts its clocking. The SCK signal must be settled to the idle state level corresponding to the selected polarity, before the SPI slave is selected by SS, else the following transaction may be desynchronized.

When the SPI slave is enabled at the hardware SS management mode, all the traffics are ignored even in case of the SS is found at active level. They are ignored until the slave detects a start of the SS signal (its transition from non-active to active level) just synchronizing the slave with the master. This is why the hardware management mode cannot be used when external SS pin is fixed. There is no such protection at the SS software management. In this case, the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

The master in full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled, the CSTART bit is set and the TxFIFO is not empty, or with the next write to TxFIFO.

In any master receive-only mode, the master starts to communicate and the clock starts running after the SPI is enabled and the CSTART bit is set.

For handling DMA, see Section 52.4.15.

52.4.12 SPI data transmission and reception procedures

The setting of data communication format follows the basic principle that sure number of data with a flexible size must be transferred within a session (transaction) while, optionally, the data handling can be cumulated effectively into a single access of the SPI data registers (data packing) or even grouped into a sequence of such services if data is collected at consistent bigger data packets. The data handling services are based upon FIFO packet occupancy events. This is why the complete data packet must be serviced exclusively upon a dedicated packet flag.
To understand better the next detailed content of this section, the user should capture the configuration impact and meaning of the following items at first:

**Data size (DSIZE)** - defines data frame (sets the number of bits at single data frame).

**FIFO threshold (FTHLV)** - defines data packet, sets the number of data frames at single data packet and so the occurrence of the packet occupancy events to handle SPI data registers either by software or by DMA.

**Data access** – a way how to handle the SPI data register content when the transfer data between the application and the SPI FIFOs upon a packet event. It depends on the packet size configuration. Optionally, multiply data can be handled effectively by a single access of the register (by data packing) or by sequence of such accesses (when servicing a bigger data packet).

**FIFO size** – capacity or space to absorb available data. It depends on the data size and the internal hardware efficiency how the data is compressed and organized within this space. The FTHLV setting must respect the FIFO capacity to store two data packets at least.

**Transaction size (TSIZE)** – defines total number of data frames involved at a transaction session overall possibly covered by several data packet services. There is no need to align this number with the packet size (handling of a last not aligned data packet is supported if TSIZE is programmed properly).

**Data handling via RxFIFO and TxFIFO**

All SPI data transactions pass through the embedded FIFOs organized by bytes (N x 8-bit). The size of the FIFOs (N) is dependent on the product and the peripheral instance. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short or the interrupt/DMA latency is too long. Each direction has its own FIFO called TxFIFO and RxFIFO, respectively.

The handling of the FIFOs content is based on servicing data packet events exclusively raised by dedicated FIFO packet occupancy flags (TXP, RXP or DXP). The flags occurrence depends on the data exchange mode (duplex, simplex), the data frame size (number of bits in the frame) and how data are organized at data packets. The frequency of the packet events can be decreased significantly when data are organized into packets via defining the FIFOs threshold. Several data frames grouped at packet can be then handled effectively based on a single FIFO occupancy packet event either by a single SPI data register access or their sequence what consumes less system performance. The user can control the access type by casting the data register address to force a concrete CPU instruction applied for the register read or write. The access then can be 8-bit, 16-bit or 32-bit but single data frame must be always accessed at least. It is crucial to keep the setting of the packet size (FTHVL) and the data size (DSIZE) always balanced with the applied data registers access (no matter if a single access or their sequence is applied) just to apply and complete service of a single data packet upon its event. This principle, occurrence and clearing capabilities of the FIFO occupancy flags are common no matter if DMA, interrupt, or polling is applied.

A read access to the SPI_RXDR register returns the oldest value stored in the RxFIFO that has not been read yet. A write access to the SPI_TXDR stores the written data in the TxFIFO at the end of a send queue.

A read access to the SPI_RXDR register must be managed by the RXP event. This flag is set by hardware when at least one complete data packet (defined as receiver threshold by FTHVL[3:0] bits at the SPI_CFG1 register) is available at the reception FIFO while reception is active. The RXP is cleared as soon as less data than complete single packet is available in the RxFIFO, when reading SPI_RXDR by software or by DMA.
The RXP triggers an interrupt if the RXPIE bit is set and/or a DMA request if the RXDMAEN bit is set.

Upon setting of the RXP flag, the application performs the due number of SPI data register reads to download the content of one data packet. Once a complete data packet is downloaded, the application software or DMA checks the RXP value to see if other packets are pending into the receive FIFO and, if so, downloads them packet by packet until the RXP reads 0. RxFIFO can store up to N data frames (for frame size ≤ 8-bit), N/2 data frames (for 8-bit < frame ≤ 16-bit), N/3 data frames (for 16-bit < frame ≤ 24-bit) or N/4 data frames (if data frame > 24-bit) where N is the size of the FIFO in bytes.

At the end of a reception, it may happen that some data are still available in the RxFIFO, without reaching the FTHLV level, thus the RXP is not set. In this case, the number of remaining RX data frames in the FIFO is indicated by RXWNE and RXPLVL fields in the SPI_SR register. It happens when the number of the last data received in a transfer cannot fully accomplish the configured packet size; in case the transfer size and the packet size are not aligned. Nevertheless the application software can still perform the standard number of reads from the RxFIFO used for the previous complete data packets without drawbacks: only the consistent data (completed data frames) are popped from the RxFIFO while redundant reads (or any uncompleted data) are reading 0. Thanks to that, the application software can treat all the data in a transfer in the same way, and is off-loaded to foresee the reception of the last data in a transfer and to calculate the due number of reads to be popped from RxFIFO.

In a similar way, the write access of a data frame to be transmitted is managed by the TXP event. This flag is set by hardware when there is enough space for the application to push at least one complete data packet (defined at FTHLV[3:0] bits at SPI_CFG1 register) into the transmission FIFO while transmission is active. The TXP is cleared as soon as the TxFIFO is filled by software and/or by the DMA. The space currently available for any next complete data packet is lost. This can lead to oscillations of the TXP signal when data are released out from the TxFIFO while a new packet is stored frame by frame. Any write to the TxFIFO is ignored when there is no sufficient room to store at least a single data frame (TXP event is not respected), when TXTF is set or when the SPI is disabled.

The TXP triggers an interrupt if the TXPIE bit is set and/or with a DMA request if the TXDMAEN bit is set. The TXPIE mask is cleared by hardware when the TXTF flag is set.

Upon setting of the TXP flag, the application performs the due number of SPI data register writes to upload the content of one entire data packet. Once new complete data packet is uploaded, the application software or DMA checks the TXP value to see if other packets can be pushed into the TxFIFO and, if so, uploads them packet by packet until TXP reads 0.

The number of last data in a transfer can be shorter than the configured packet size in the case when the transfer size and the packet size are not aligned. Nevertheless the application software can still perform the standard number of data register writes used for the previous packets without drawbacks: only the consistent data are pushed into the TxFIFO while redundant writes are discarded. Thanks to that, the application software can treat all the data in a transfer in the same way and is off-loaded to foresee the transmission of the last data in a transfer and from calculating the due number of writes to push the last data into TxFIFO. Just for the last data case, the TXP event is asserted by SPI once there is enough space into TxFIFO to store remaining data to complete current transfer.

Both TXP and RXP events can be polled or handled by interrupts. The DXP bit can be monitored as a common TXP and RXP event at full-duplex mode.
Upon setting of the DXP flag the application performs the due number of writes to the SPI data register to upload the content of one entire data packet for transmission, followed by the same number of reads from the SPI data register to download the content of one data packet. Once one data packet is uploaded and one is downloaded, the application software or DMA checks the DXP value to see if other packets can be pushed and popped in sequence and, if so, uploads/downloads them packet by packet until DXP reads 0.

The DXP triggers an interrupt if the DXPIE bit is set. The DXPIE mask is cleared by hardware when the TXTF flag is set.

The DXP is useful in full-duplex communication in order to optimize performance in data uploading/downloading, and reducing the number of interrupts or DMA sequences required to support an SPI transfer thus minimizing the request for CPU bandwidth and system power especially when SPI is operated in Stop mode.

When relay on the DXP interrupt exclusively, the user must consider the drawback of such a simplification when TXP and RXP events are serviced by common procedures because the TXP services are delayed by purpose in this case. This is due to fact that the TXP events precedes the reception RXP ones normally to allow the TXP servicing prior transaction of the last frame fully emptying the TxFIFO else master cannot provide a continuous SCK clock flow and the slave can even face an underrun condition. The possible solution is to prefill the TxFIFO by few data packets ahead prior the session starts and to handle all the data received after the TXTF event by EOT exclusively at the end of the transaction (as TXTF suppresses the DXP interrupts at the end of the transaction). In case of CRC computation is enabled, the user must calculate with additional space to accommodate the CRC frame at RxFIFO when relying on EOT exclusively at the end of transaction.

Another way to manage the data exchange is to use DMA (see Section 52.4.15).

If the next data is received when the RxFIFO is full, an overrun event occurs (see description of OVR flag in Section 52.5.2). An overrun event can be polled or handled by an interrupt.

This may happen in Slave mode or in a master receive mode when MASRX = 0. If MASRX bit is set at a master receiver, the generated clock stops automatically when the RxFIFO is full, therefore overrun is prevented.

Both RxFIFO and TxFIFO content is kept flushed and cannot be accessed when SPI is disabled (SPE = 0).

Transaction handling

A few data frames can be passed at single sequence to complete a message. The user can handle a number of data within a message thanks to the value stored into TSIZE. In principle, the transaction of a message starts when the SPI is enabled by setting CSTART bit and finishes when the total number of required data is transacted. The end of transaction controls the CRC and the hardware SS management when applied. To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transaction starts despite its setting is not changed.

If TSIZE is kept at zero while CSTART is set, an endless transaction is initialized (no control of transfer size is applied). During an endless transaction, the number of transacted data aligned with FIFOs threshold is supported exclusively. If the number of data (or its grouping into packets) is unpredictable, the user must keep the FIFO threshold setting (packet size) at single data (FTHLV = 0) to assure that each data frame raises its own packet event to be serviced by the application or DMA. The transaction can be suspended at any time thanks
to CSUSP which clears the CSTART bit. SPI must be always disabled after such software suspension and re-enabled before the next transaction starts.

When the transmission is enabled, a sequence begins and continues while any data is present in the TxFIFO of the master. The clock signal is provided permanently by the master until TxFIFO becomes empty, then it stops, waiting for additional data.

In receive-only modes, half-duplex (COMM[1:0] = 11, HDDIR = 0) or simplex (COMM[1:0] = 10) modes, the master starts the sequence when the SPI is enabled and the transaction is released by setting the CSTART bit. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled/suspended by the master. The master receives data frames permanently up to this moment. The reception can be suspended either by software control, writing 1 to the CSUSP bit in the SPI_CR1 register, or automatically when MASRX = 1 and RxFIFO becomes full or upon the RDY status if this signal is applied (see Section 52.4.8). The reception is automatically stopped also when the number of frames programmed in TSIZE has been completed.

In order to disable the master receive-only mode, the SPI must first be suspended. When the SPI is suspended, the current frame is completed, before changing the configuration.

**Caution:** If the SPE bit is cleared in Master mode, while the reception is ongoing without any suspending, the clock is stopped without completing the current frame, and the RxFIFO is flushed.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it must respect the slave capability to handle data flow and its content at anytime. If the slave features the RDY signal option, the master can monitor the RDY signal issued by the slave, to control the communication flow. If the RDY pin is not used, the slave is considered always ready for communication with the master.

When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays by MIDI[3:0] bits setting or provide an initial delay by setting MSS[1:0], which postpones any transaction start to give slave sufficient room for preparing data. Be aware data from the slave are always transacted and processed by the master even if the slave cannot prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are short, FIFO is accessed by bytes and the SPI bus rate is high.

In order to add some software control on the SPI communication flow from a slave transmitter node, a specific value written in the SPI_UDRDR (SPI Underrun Data Register) may be used. On slave side, when TxFIFO becomes empty, this value is sent out automatically as next data and may be interpreted by software on the master receiver side (either simply dropped or interpreted as a XOFF like command, in order to suspend the master receiver by software).

In the multislave star topology, only a single slave only can be enabled for output data at a given time. The slave just selected for the communication with the master needs to detect a change of its SS input into active level before communication with the master starts. In a single slave system it is not necessary to control the slave with SS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. The SS can be managed by both software and hardware (Section 52.4.7).

### 52.4.13 Disabling the SPI

To disable the SPI, it is mandatory to follow the disable procedures described in this paragraph.
In the Master mode, it is important to do this before the system enters a low-power mode when the peripheral clock is stopped, otherwise, ongoing transactions may be corrupted.

In Slave mode, the SPI communication can continue when the \texttt{spi\_pclk} and \texttt{spi\_ker\_ck} clocks are stopped, without interruption, until any end of communication or data service request condition is reached. The \texttt{spi\_pclk} can generally be stopped by setting the system into Stop mode. Refer to the RCC section for further information.

The master in full-duplex or transmit-only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. TXC flag can be polled (or interrupt enabled with EOTIE = 1) in order to wait for the last data frame to be sent.

When the master is in any receive-only mode, in order to stop the peripheral, the SPI communication must first be suspended, by setting the CSUSP bit to 1.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

After such a software suspension, SPI must be always disabled to restart the internal state machine properly.

When SPI is disabled, RxFIFO is flushed. To prevent losing unread data, the user must ensure that RxFIFO is empty when disabling the SPI, by reading all remaining data (as indicated by the RXP, RXWNE and RXPLVL fields in the SPI\_SR register).

The standard disable procedure is based on polling EOT and/or TXC status to check if a transmission session is (fully) completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When the master handles SS signal by a GPIO not related to SPI (for example at case of multislave star topology) and it has to provide proper end of SS pulse for slave, or
- When transaction streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

When TSIZE>0, EOT and TXC signals are equal so polling of EOT is reliable at whatever SPI communication mode to check end of the bus activity. When TSIZE = 0, the user has to check TXC, SUSP or FIFO occupancy flags in according with the applied SPI mode and the way of the data flow termination.

The correct disable procedure in Master mode, except when receive-only mode is used, is:

1. Wait until TXC = 1 and/or EOT = 1 (no more data to transmit and last data frame sent). When CRC is used, it is sent automatically after the last data in the block is processed. TXC/EOT is set when CRC frame is completed in this case. When a transmission is suspended the software has to wait till CSTART bit is cleared.
2. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
3. Disable the SPI (SPE = 0).

The correct disable procedure for master receive-only modes is:

1. Wait on EOT or break the receive flow by suspending SPI (CSUSP = 1).
2. Wait until SUSP = 1 (the last data frame is processed) if receive flow is suspended.
3. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
4. Disable the SPI (SPE = 0).

In Slave mode, any on going data are lost when disabling the SPI.
Controlling the I/Os

As soon as the SPI is disabled, the associated and enabled AF outputs can still be driven by the device depending on the AFCNTR setting. When active output control is applied (AFCNTR = 1) and SPI is just been disabled (SPE = 0), the enabled outputs associated with SPI control signals (like SS and SCK at master and RDY at slave) can toggle immediately to inactive level (according to SSIOP and CPOL settings at master and RDIOP at slave respectively). The data line output (MOSI at master and MISO at slave) can instead change its level immediately at dependency on the actual TxFIFO content with the effect of potentially making invalid and no more guaranteed the value of the latest transacted bit on the bus. If necessary, the user has to take care about proper data hold time at the data line and avoid any eventual fast SPI disable just after the last data transaction is completed.

Note: Despite stability of the latest bit is guaranteed by design during the sampling edge of the clock, some devices can require even extension of this data bit stability interval during the sampling. It can be done for example by inserting small software delay between EOT event occurrence and SPI disable action.

52.4.14 Data packing

From user point of view there are two ways of data packing which can overlay each other:

- Type of access when data are written to TxFIFO or read from RxFIFO
  Multiple data can be pushed or fetched effectively by single access if data size is multiplied less than the access performed upon SPI_TXDR or SPI_RXDR registers.
- Number of data to be handled during the single software service
  It is convenient to group data into packets and cumulate the FIFO services overall the data packet content exclusively instead of handling data frame by frame separately. The user can define packets by FIFO threshold settings. Then all the FIFO occupancy events are related to that threshold level while required services are signalized by proper flags with interrupt and/or wake up capabilities.

When the data frame size fits into one byte (less than or equal to 8 bits), the data packing is used automatically when any read or write 16-bit or 32-bit access is performed on the SPI_RXDR/SPI_TXDR register. The multiple data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other data stored in the MSB.

Figure 727 provides an example of data packing mode sequence handling at full feature set instance. While DSIZE[4:0] is configured to 4-bit there, two or four data frames are written in the TxFIFO after the single 16-bit or 32-bit access the SPI_TXDR register of the transmitter. When the data frame size is between 9-bit and 16-bit, data packing is used automatically when a 32-bit access is done. Least significant half-word is used first. (regardless of the LSBFRST value)

This sequence can generate two or four RXP events in the receiver if the RxFIFO threshold is set frame (and data is read on a frame basis, unpacked), or it can generate a single RXP event if the FTHLV[3:0] field in the SPI_CFG1 register is programmed to a multiple of the frames to be read in a packed mode (16-bit or 32-bit read access).

The data are aligned in accordance with Figure 726. The valid bits are performed on the bus exclusively. Unused bits are not cared at transmitter while padded by zeros at receiver.

When short data frames (< 8-bit or < 16-bit) are used together with a larger data access mode (16-bit or 32-bit), the FTHLV value must be programmed as a multiple of the number...
of frames/data access (multiple of 4 if 32-bit access is used to up to 8-bit frames or multiple of 2 if 16-bit access is used to up to 8-bit frames or 32-bit access to up to 16-bit frames.).

The RxFIFO threshold setting must always be higher or equal at least than the following read access size, as spurious extra data would be read otherwise.

The FIFO data access less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

A specific problem appears if an incomplete data packet is available at FIFO: less than threshold set at FTHLV bits.

There are two ways of dealing with this problem:

A. without using TSIZE field

On transmitter side, writing the last data frame of any odd sequence with an 8-bit/16-bit access to SPI_TXDR is enough.

On receiver side, the remaining data may be read by any access. Any extra data read are padded with zeros. Polling the RXWNE and RXPLVL may be used to detect when the RX data are available in the RxFIFO. (A time out may be used at system level in order to detect the polling)

B. using the TSIZE field

On transmitter side, the transaction is stopped by the master when it faces EOT event.

In reception, the RXP flag is not set when EOT is set. In the case when the number of data to be received (TSIZE) is not a multiple of packet size, the number of remaining data is indicated by the RXWNE and RXPLVL fields in the SPI_SR register. The remaining data can be read by any access. Any extra read is padded by zeros.

Figure 727. Packing data in FIFO for transmission and reception at full feature set instance

1. DSIZE[4:0] is configured to 4-bit, data is right aligned, valid bits are performed only on the bus, their order depends on LSBFRST, if it is set, the order is reversed at all the data frames.

52.4.15 Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.
A DMA access is requested when the TXDMAEN or RXDMAEN enable bits in the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers to fulfill service of the defined packet.

- In transmission, a series of DMA requests is triggered each time TXP is set to 1. The DMA then performs series of writes to the SPI_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set to 1. The DMA then performs series of reads from the SPI_RXDR register. When EOT is set at the end of transaction and last data packet is incomplete then DMA request is activated automatically in according with RXWNE and RXPLVL[1:0] setting to read rest of data.

If the SPI is programmed in receive-only mode, UDR is never set.

If the SPI is programmed in a transmit mode, TXP and UDR can be eventually set at slave side, because transmit data may not be available. In this case, some data are sent on the TX line according with the UDR management selection.

When the SPI is used at a simplex mode, the user must enable the adequate DMA channel only while keeping the complementary unused channel disabled.

If the SPI is programmed in transmit-only mode, RXP and OVR are never set.

If the SPI is programmed in full-duplex mode, RXP and OVR are eventually set, because received data are not read.

In transmission mode, when the DMA or the user has written all the data to be transmitted (the TXTF flag is set at SPI_SR register), the EOT (or TXC at case TSIZE = 0) flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or before disabling the spi_pclk in Master mode. The software must first wait until EOT = 1 and/or TXC = 1.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:
1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI_CFG1 register, if DMA Rx is used.
2. Enable DMA requests for Tx and Rx in DMA registers, if the DMA is used.
3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI_CFG1 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication it is mandatory to follow these steps in order:
1. Disable DMA request for Tx and Rx in the DMA registers, if the DMA issued.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits in the SPI_CFG1 register, if DMA Tx and/or DMA Rx are used.

Data packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPI_CFG1 register) the packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel.

If the DMA channel PSIZE value is equal to 16-bit and the SPI data size is less than or equal to 8-bit, then the packing mode is enabled. Similarly, If the DMA channel PSIZE value is equal to 32-bit and the SPI data size is less than or equal to 16-bit, then the packing mode is
enabled. The DMA then automatically manages the write operations to the SPI_TXDR register.

Regardless data packing mode is used and the number of data to transfer is not a multiple of the DMA data size (16-bit or 32-bit) while the frame size is smaller, DMA completes the transfer automatically in according with the TSIZE field setting.

Alternatively, last data frames may be written by software, in the single/unpacked mode.

To configure any DMA data access less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

52.5  SPI specific modes and control

52.5.1  TI mode

With a specific SP[2:0] bit field setting of the SPI_CFG2 register, the SPI can be configured compliant with the TI protocol. The SCK and SS signals polarity, phase and flow as well as the bits order are fixed so the setting of CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI and MIDI is not required when the SPI is in TI mode configuration. The SS signal synchronizes the protocol by pulses over the LSB data bit as it is shown in Figure 728.

![Figure 728. TI mode transfer](image)

In Slave mode, the clock generator is used to define time when the slave output at MISO pin becomes to HiZ when the current transaction finishes. The master baud rate setting (MBR[2:0] at SPI_CFG1) is applied and any baud rate can be used to determine this moment with optimal flexibility. The delay for the MISO signal to become HiZ (TRELEASE) depends on internal re-synchronization, too, which takes next additional 2-4 periods of the clock signal feeding the generator. It is given by formula:

\[
\frac{T_{baud}}{2} + 2 \times T_{spi\_ker\_ck} \leq T_{release} \leq \frac{T_{baud}}{2} + 4 \times T_{spi\_ker\_ck}
\]

If the slave detects misplaced SS pulse during data transaction the TIFRE flag is set.
52.5.2 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the corresponding Interrupt Enable bit.

**Overrun flag (OVR)**

An overrun condition occurs when data are received by a master or slave and the RxFIFO has not enough space to store these received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RxFIFO).

When an overrun condition occurs, the OVR flag is set and the newly received value does not overwrite the previous one in the RxFIFO. The newly received value is discarded and all data transmitted subsequently are lost. OVR flag triggers an interrupt if OVRIE bit is set.

Clearing the OVR bit is done by a writing 1 to the OVRC bit in the SPI_IFCR. To prevent any next overrun event the clearing should be done after RxFIFO is emptied by software reads. It is suggested to release the RxFIFO space as much as possible, this means to read out all the available data packets based on RXP flag indication.

In Master mode, the user can prevent the RxFIFO overrun by automatic communication suspend (MASRX bit).

**Underrun flag (UDR)**

In a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO commonly. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

Underrun detection logic and system behavior depends on the UDRCFG bit. When an underrun is detected by slave, it can provide out either a constant pattern stored by the user at the UDRDR register or the data received previously from the master. When the first configuration (UDRCFG = 0) is applied, the underrun condition is evaluated whenever master starts to communicate a new data frame while TxFIFO is empty. Then single additional dummy (accidental) data is always inserted between last valid data and constant pattern defined at the UDRDR register (see Figure 729). The second configuration (UDRCFG = 1) can be used at circular topography structure (see Figure 720). Assuming that TxFIFO is not empty when master starts the communication, the underrun condition is evaluated just once the FIFO becomes empty during the next data flow. Valid data from TxFIFO is then upended by the lastly received data immediately.

The standard transmission is re-enabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. The user should write some data into TxFIFO prior clearing the UDR flag to prevent any next underrun condition occurrence capture.

The data transacted by slave is unpredictable especially when the transaction starts or continues while TxFIFO is empty and underrun condition is either not yet captured or just cleared. Typically, this is the case when SPI is just enabled or when a transaction with a defined size just starts. First bits can be corrupted in this case, as well, when slave software writes first data into the empty TxFIFO too close prior the data transaction starts (propagation of the data into TxFIFO takes few APB clock cycles).
Figure 729. Optional configurations of slave detecting underrun condition

Note: The hardware propagation of an UDR event needs additional traffic on the bus. It always takes few extra SPI clock cycles after the event happens (both underrun captured by hardware and cleared by software). If clearing of the UDR flag by software is applied close to the end of data frame transaction or when SCK line is at idle in between the frames, next extra underrun pattern is sent initially by slave prior the valid data from Tx FIFO becomes transacted again. The user can prevent this by SPI disable/enable action between sessions to restart the underrun logic and so initiate the next session by the valid data.

Mode fault (MODF)

Mode fault occurs when the master device has its internal SS signal (SS pin in SS hardware mode, or SSI bit in SS software mode) pulled low. This automatically affects the SPI interface in the following ways:

- The MODF bit is set and the SPI interrupt is triggered if the MODFIE bit is set.
- The SPE bit is forced to zero till MODF bit is set. This disables SPI and blocks all the peripheral outputs except the MODF interrupt request if enabled.
- The MASTER bit is cleared, thus forcing the device into Slave mode.

MODF is cleared by writing 1 to the MODFC bit in the SPI_IFCR.

To avoid any multiple slave conflicts in a system comprising several MCUs, the SS pin must be pulled to its non-active level before re-enabling the SPI, by setting the SPE bit.
As a security, hardware does not allow the SPE bit to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multi master conflict.

A correct software procedure when master overtakes the bus at multi master system should be the following one:

- Switch into Master mode while SSOE = 0 (potential conflict can appear when another master occupies the bus. MODF is raised in this case, preventing any next node switching into Master mode)
- Put GPIO pin dedicated for another master SS control into active level
- Perform data transaction
- Put GPIO pin dedicated for another master SS control into non active level
- Switch back to Slave mode

**CRC error (CRCE)**

This flag is used to verify the validity of the value received when the CRCEN bit in the SPI_CFG1 register is set. The CRCE flag in the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRC value, after the last data is received (as defined by TSIZE). The CRCE flag triggers an interrupt if CRCEIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit in the SPI_IFCR.

**TI mode frame format error (TIFRE)**

A TI mode frame format error is detected when an SS pulse occurs during an ongoing communication when the SPI is operating in Slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the SS pulse is ignored, and the SPI waits for the next SS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of few data frames.

The TIFRE flag is cleared by writing 1 to the TIFREC bit in the SPI_IFCR. If the TIFREIE bit is set, an interrupt is generated on the SS error detection. As data consistency is no longer guaranteed, communication should be re-initiated by software between master and slave.

### 52.5.3 CRC computation

Two separate 33-bit or two separate 17-bit CRC calculators are implemented to check the reliability of transmitted and received data. For instances with full feature set, the SPI offers CRC polynomial length from 5 to 33 bits when maximum data size is 32-bit and from 9 to 17 bits for the peripheral instances with data size limited to 16 bits. For instances with limited set of features, the CRC polynomial length can be set either to 9 or 17 only when data size is limited to 16 bit and optionally to 33 when data size is extended to 32-bit. The length of the polynomial is defined by the most significant bit of the value stored at the CRCPOLY register. It must be set greater than data frame length defined at DSIZE field. When maximum data size is applied, the CRC33_17 bit must be set additionally to define the most significant bit of the polynomial string while keep its size always greater than data. The CRCSIZE field in the SPI_CFG1 then defines how many the most significant bits from CRC calculation registers are transacted and compared as CRC frame. It is defined independently from the data frame length, but it must be either equal or an integer multiple of the data frame size while its size cannot exceed the maximum data size of the instance.

To fully benefit from the CRC calculation capability, the polynomial length setting must correspond to the CRC pattern size, else the bits unused at the calculation are transacted...
and expected all zero at the end of the CRC pattern if its size is set greater then the polynomial length.

**CRC principle**

The CRC calculation is enabled by setting the CRCEN bit in the SPI_CFG1 register before the SPI is enabled (SPE = 1). The CRC value is then calculated using the CRC polynomial defined by the CRCPOLY register and CRC33_17 bit. When SPI is enabled, the CRC polynomial can be changed but only in case when there is no traffic on the bus.

The CRC computation is done, bit by bit, on the sampling clock edge defined by the CPHA and CPOL bits in the SPI_CR1 register. The calculated CRC value is checked automatically at the end of the data block defined by the SPI_CR2 register exclusively.

When a mismatch is detected between the CRC calculated internally on the received data and the CRC received from the transmitter, a CRCE flag is set to indicate a data corruption error. The right procedure for handling the CRC depends on the SPI configuration and the chosen transfer management.

**CRC transfer management**

Communication starts and continues normally until the last data frame must be sent or received in the SPI_DR register.

The length of the transfer must be defined by TSIZE. When the desired number of data is transacted, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

No matter what is the CRCSIZE configuration, TSIZE cannot be set neither to 0xFFFF at full feature set instance nor to 0x3FF value at limited feature one if CRC is enabled.

In transmission, the CRC computation is frozen during CRC transaction and the TXCRC are transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when desired number of data is transacted. Information to be compared with the RXCRC register content is then received in a frame of length equal to the CRCSIZE value.

Once the CRC frame is completed, an automatic check is performed comparing the received CRC value and the value calculated in the SPI_RXCRC register. The software has to check the CRCE flag in the SPI_SR register to determine if the data transfers were corrupted or not. Software clears the CRCE flag by writing 1 to the CRCEC.

The user takes no care about any flushing redundant CRC information, it is done automatically.

**Resetting the SPI_TXCRC and SPI_RXCRC values**

The SPI_TXCRC and SPI_RXCRC values are initialized automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode in order to transfer data without any interruption (several data blocks covered by intermediate CRC checking phases). Initialization patterns for receiver and transmitter can be configured either to zero or to all ones in dependency on setting bits TCRCINI and RCRCINI at SPI_CR1 register.

The CRC values are reset when the SPI is disabled.
52.6 SPI low-power modes

Table 555. Effect of low-power modes on the SPI

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. SPI interrupts cause the device to exit Sleep mode.</td>
</tr>
<tr>
<td>Stop(1)</td>
<td>The SPI registers content is kept.</td>
</tr>
<tr>
<td>Standby(1)</td>
<td>The SPI instance not functional when this mode is powered down and must be</td>
</tr>
<tr>
<td></td>
<td>reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

1. Refer to Section 52.3: SPI implementation for information about wake-up from Stop mode support per instance as well as Standby mode availability. If an instance is not functional in a Stop mode, it must be disabled before entering this Stop mode.

52.7 SPI interrupts

Table 556 gives an overview of the SPI events capable to generate interrupts if enabled. Some of them feature wake up from Low-power mode capability, additionally. Most of them can be enabled and disabled independently while using specific interrupt enable control bits. The flags associated with the events are cleared by specific methods. Refer to description of SPI registers for more details about the event flags. When SPI is disabled, all the pending interrupt requests are blocked to prevent their propagation into the interrupt services, except the MODF interrupt request.

Table 556. SPI wake-up and interrupt requests

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Event clear method</th>
<th>Exit from Stop and Standby modes capability(1)(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI</td>
<td>TxFIFO ready to be loaded (space available for one data packet - FIFO threshold)</td>
<td>TXP</td>
<td>TXPIE</td>
<td>TXP cleared by hardware when TxFIFO contains less than FTHLV empty locations</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Data received in RxFIFO (one data packet available - FIFO threshold)</td>
<td>RXP</td>
<td>RXPIE</td>
<td>RXP cleared by hardware when RxFIFO contains less than FTHLV samples</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Both TXP and RXP active</td>
<td>DXP</td>
<td>DXPIE</td>
<td>When TXP or RXP are cleared</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Transmission Transfer Filled</td>
<td>TXTF</td>
<td>TXTFIE</td>
<td>Writing TXTFC to 1</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Underrun</td>
<td>UDR</td>
<td>UDRIE</td>
<td>Writing UDRC to 1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Overrun</td>
<td>OVR</td>
<td>OVRIE</td>
<td>Writing OVRC to 1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CRC Error</td>
<td>CRCE</td>
<td>CRCEIE</td>
<td>Writing CRCEC to 1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>TI Frame Format Error</td>
<td>TIFRE</td>
<td>TIFREIE</td>
<td>Writing TIFREC to 1</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Mode Fault</td>
<td>MODF</td>
<td>MODFIE</td>
<td>Writing MODFC to 1</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>End Of Transfer (full transfer sequence completed - based on TSIZE value)</td>
<td>EOT</td>
<td>EOTIE</td>
<td>Writing EOTC to 1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Master mode suspended</td>
<td>SUSP</td>
<td>EOTIE</td>
<td>Writing SUSPC to 1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>TxFIFO transmission complete (TxFIFO empty)</td>
<td>TXC(3)</td>
<td></td>
<td>TXC cleared by hardware when a transmission activity starts on the bus</td>
<td>No</td>
</tr>
</tbody>
</table>
52.8 I2S main features

- Full duplex communication
- Simplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler
- Data length may be 16, 24 or 32 bits\(^{a}\)
- Channel length can be 16 or 32 in master, any value in slave
- Programmable clock polarity
- Error flags signaling for improved reliability: Underrun, Overrun and Frame Error
- Embedded Rx and TxFIFOs
- Supported I2S protocols:
  - I2S Philips standard
  - MSB-Justified standard (Left-Justified)
  - LSB-Justified standard (Right-Justified)
  - PCM standard (with short and long frame synchronization)
- Data ordering programmable (LSb or MSb first)
- DMA capability for transmission and reception
- Master clock can be output to drive an external audio component:
  - \(F_{MCK} = 256 \times F_{WS}\) for all I2S modes
  - \(F_{MCK} = 128 \times F_{WS}\) for all PCM modes

Note: \(F_{MCK}\) is the master clock frequency and \(F_{WS}\) is the audio sampling frequency.

52.9 I2S functional description

52.9.1 I2S general description

The block diagram shown on Figure 715 also applies for I2S mode.

The SPI/I2S block can work on I2S/PCM mode, when the bit I2SMOD is set. A dedicated register (SPI_I2SCFGR) is available for configuring the dedicated I2S parameters, which include the clock generator, and the serial link interface.

---

\(^{a}\) Not always available, refer to Section 52.3: SPI implementation in order to check if 24 and 32-bit data width are supported.
The I2S/PCM function uses the clock generator to produce the communication clock when the SPI/I2S is set in master mode. This clock generator is also the source of the master clock output (MCK).

Resources such as RxFIFO, TxFIFO, DMA and parts of interrupt signaling are shared with SPI function. The low-power mode function is also available in I2S mode, refer to Section 52.6: SPI low-power modes and Section 52.10: I2S interrupts.

52.9.2 Pin sharing with SPI function

The I2S shares four common pins with the SPI:
- SDO: Serial Data Output (mapped on the MOSI pin) to transmit the audio samples in master, and to receive the audio sample in slave. Refer to Section : Serial data line swapping on page 2414.
- SDI: Serial Data Input (mapped on the MISO pin) to receive the audio samples in master, and to transmit the audio sample in slave. Refer to Section : Serial data line swapping on page 2414.
- WS: Word Select (mapped on the SS pin) is the frame synchronization. It is configured as output in master mode, and as input for slave mode.
- CK: Serial Clock (mapped on the SCK pin) is the serial bit clock. It is configured as output in master mode, and as input for slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:
- MCK: Master Clock (mapped separately) is used when the I2S is configured in master mode.

52.9.3 Bitfields usable in I2S/PCM mode

When the I2S/PCM mode is selected (I2SMOD = '1'), some bitfields are no longer relevant, and must be forced to a specific value in order to guarantee the behavior of the I2S/PCM function. Table 557 shows the list of bits and fields available in the I2S/PCM mode, and indicates which must be forced to a specific value.

<table>
<thead>
<tr>
<th>Register name</th>
<th>Bitfields usable in PCM/I2S Mode</th>
<th>Constraints on other bitfields</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI/I2S control register 1 (SPI_CR1)</td>
<td>IOLOCK, CSUSP, CSTART, SPE</td>
<td>Other fields set to their reset values</td>
</tr>
<tr>
<td>SPI/I2S control register 2 (SPI_CR2)</td>
<td>-</td>
<td>Set to reset value</td>
</tr>
<tr>
<td>SPI/I2S configuration register 1 (SPI_CFG1)</td>
<td>TXDMAEN, RXDMAEN, FTHLV</td>
<td>Other fields set to their reset values</td>
</tr>
<tr>
<td>SPI/I2S configuration register 2 (SPI_CFG2)</td>
<td>AFCNTR, LSBFRST, IOSWP</td>
<td>Other fields set to their reset values</td>
</tr>
<tr>
<td>SPI/I2S interrupt enable register (SPI_IER)</td>
<td>TIFREIE, OVRIE, UDRIE, TXPIE, RXPIE</td>
<td>Other fields set to their reset values</td>
</tr>
<tr>
<td>SPI/I2S status register (SPI_SR)</td>
<td>SUSP, TIFRE, OVR, UDR, TXP, RXP</td>
<td>Other flags not relevant</td>
</tr>
</tbody>
</table>
52.9.4 Slave and master modes

The SPI/I2S block supports master and slave mode for both I2S and PCM protocols. In master mode, both CK, WS and MCK signals are set to output. In slave mode, both CK and WS signals are set to input. The signal MCK cannot be used in slave mode.

In order to improve the robustness of the SPI/I2S block in slave mode, the peripheral re-synchronizes each reception and transmission on WS signal. This means that:

- In I2S Philips standard, the shift-in or shift-out of each data is triggered one bit clock after each transition of WS.
- In I2S MSB justified standard, the shift-in or shift-out of each data is triggered as soon as a transition of WS is detected.
- In PCM short standard, the shift-in or shift-out of each data is triggered one bit clock after the active edge of WS.
- In PCM long standard, the shift-in or shift-out of each data is triggered as soon as the active edge of WS is detected.

Note: This re-synchronization mechanism is not available for the I2S LSB justified standard.

Note as well that there is no need to provide a kernel clock when the SPI/I2S is configured in slave mode.

52.9.5 Supported audio protocols

The I2S/PCM interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPI_I2SCFGR register.

In the I2S protocol, the audio data are time-multiplexed on two channels: the left channel and the right channel. The WS signal is used to indicate which channel must be considered as the left, and which one is the right.

In I2S master mode, four frames formats are supported:

- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel
- 32-bit data packed in a 32-bit channel

**Table 557. Bitfields usable in PCM/I2S mode (continued)**

<table>
<thead>
<tr>
<th>Register name</th>
<th>Bitfields usable in PCM/I2S Mode</th>
<th>Constraints on other bitfields</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI/I2S interrupt/status flags clear register (SPI_IFCR)</td>
<td>SUSPC, TIFREC, OVRC, UDRC</td>
<td>Other fields set to their reset values</td>
</tr>
<tr>
<td>SPI/I2S receive data register (SPI_RXDR)</td>
<td>The complete register</td>
<td></td>
</tr>
<tr>
<td>SPI/I2S polynomial register (SPI_CRCPOLY)</td>
<td>-</td>
<td>Set to reset value</td>
</tr>
<tr>
<td>SPI/I2S transmitter CRC register (SPI_TXCRC)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SPI/I2S receiver CRC register (SPI_RXCRC)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SPI/I2S underrun data register (SPI_UDRDR)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SPI/I2S configuration register (SPI_I2SCFGR)</td>
<td>The complete register</td>
<td></td>
</tr>
</tbody>
</table>
In PCM master mode, three frames formats are supported:

- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel\(^{(a)}\)

The figure hereafter shows the main definition used in this section: data length, channel length and frame length.

**Figure 730. Waveform examples**

For simplicity sake, in the next figures, SDI represents the serial data input and SDO the serial data output. Refer to Section 52.3: SPI implementation in order to check if 24 and 32-bit data width are supported.

**I^2S Philips standard**

The I^2S Philips standard is selected by setting I2SSTD to 0b00. This standard is supported in master and slave mode.

In this standard, the WS signal toggles one CK clock cycle before the first bit (MSb in I^2S Philips standard) is available. A falling edge transition of WS indicates that the next data transferred is the left channel, and a rising edge transition indicates that the next data transferred is the right channel.

---

\(a\). Not always available, refer to Section 52.3: SPI implementation in order to check if 24 and 32-bit data width are supported.
1. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10). Refer to Section 52.3: SPI implementation for the supported data sizes.

CKPOL is cleared in order to match the I2S Philips protocol. See Selection of the CK sampling edge for information concerning the handling of WS signal.

Figure 731 shows an example of waveform generated by the SPI/I2S in the case where the channel length is equal to the data length. More precisely, this is true when CHLEN = 0 and DATLEN = 0b00 or when CHLEN = 1 and DATLEN = 0b10.

See Control of the WS Inversion for information concerning the handling of WS signal.

1. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10). Refer to Section 52.3: SPI implementation for the supported data sizes.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in transmit mode. This is applicable for both master and slave mode.
MSB justified standard

For this standard, the WS signal toggles when the first data bit is provided. The data transferred represents the left channel if WS is high, and the right channel if WS is low.

Figure 733. Master MSB Justified 16-bit or 32-bit full-accuracy length

1. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), Refer to Section 52.3: SPI implementation to check the supported data size.

CKPOL is cleared in order to match the I2S MSB justified protocol. See Selection of the CK sampling edge for information concerning the handling of WS signal.

See Control of the WS Inversion for information concerning the handling of WS signal.

Figure 734. Master MSB justified 16 or 24-bit data length

1. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), Refer to Section 52.3: SPI implementation to check the supported data size.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in master transmit mode. In slave transmit the remaining bits are forced to the value of the first bit of the next data to be generated in order to avoid timing issues (see Figure 735).
1. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), Refer to Section 52.3: SPI implementation to check the supported data size.

**LSB justified standard**

This standard is similar to the MSB justified standard in master mode (no difference for the 16 and 32-bit full-accuracy frame formats). The LSB justified 16 or 32-bit full-accuracy format give similar waveforms than MSB justified mode (see Figure 733) because the channel and data have the same length.

*Note: In the LSB justified format, only 16 and 32-bit channel length are supported in master and slave mode. This is due to the fact that it is not possible to transfer properly the data if the channel length is not known by transmitter and receiver side.*

**Figure 736. LSB justified 16 or 24-bit data length**

1. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), Refer to Section 52.3: SPI implementation to check the supported data size.

CKPOL is cleared in order to match the I2S LSB justified protocol. See Selection of the CK sampling edge for information concerning the handling of WS signal.

See Control of the WS Inversion for information concerning the handling of WS signal.
PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and can be selected using the PCMSYNC bit in SPI_I2SCFGR register.

In PCM long frame:
- The assertion time of WS signal is fixed to 13 cycles of CK in master mode,
- The first data bit is received or transmitted as soon as the WS signal is asserted.

In PCM short frame:
- The assertion time of WS signal is fixed to one cycle of CK in master mode,
- The first data bit is received or transmitted one cycle of CK after the WS assertion.

For both PCM modes:
- The first data bit is MSb or LSb depending on LSBFIRST bit value.
- The CK sampling edge can be selected thanks to CKPOL bit.
- The WS signal can be inverted thanks to WSINV bit. See Control of the WS Inversion for information concerning the handling of WS signal.

Figure 737. Master PCM when the frame length is equal the data length

A data size of 16 or 24 bits can be used when the channel length is set to 32 bits.

Figure 738. Master PCM standard waveforms (16 or 24-bit data length)

1. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), Refer to Section 52.3: SPI Implementation to check the supported data size.
If the PCM protocol is used in slave mode, frame lengths can be different from 16 or 32 bits. As shown in Figure 739, in slave mode various pulse widths of WS can be accepted as the start of frame is detected by a rising edge of WS. The only constraint is that the WS must go back to its inactive state for at least one CK cycle.

**Figure 739. Slave PCM waveforms**

1. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10). Refer to Section 52.3: SPI implementation to check the supported data size.

**Note:** In the case where the channel length is bigger than the data length, in slave PCM long, the transmission of the remaining bits are forced to the value of the first bit of the next data to be generated if the TXFIFO contains the next data to be transmitted.

**Note:** In slave mode, CHLEN must be always programmed properly (whatever FIXCH value). For example, if CHLEN is cleared (16-bit length), the data transfer is truncated to 16 bits even if DATLEN is different from 0. To avoid this situation, CHLEN must be set, if DATLEN = 1 or 2.

### 52.9.6 Additional serial interface flexibility

#### Variable frame length in slave

In slave mode, channel lengths different from 16 or 32 bits can be accepted, as long as the channel length is bigger than the data length. This is true for all protocols except for I2S LSB justified protocol.
Data ordering

For all data formats and communication standards, it is possible to select the data ordering (MSb or LSb first) thanks to the bit LSBFRST located into SPI/I2S configuration register 2 (SPI_CFG2).

Selection of the CK sampling edge

The CKPOL bit located into SPI/I2S configuration register (SPI_I2SCFGR) allows the user to choose the sampling edge polarity of the CK for slave and master modes, for all protocols.

- When CKPOL = 0, serial data SDO and WS (when master) are changed on the falling edge of CK and the serial data SDI and WS (when slave) are read on the rising edge.
- When CKPOL = 1, serial data SDO and WS (when master) are changed on the rising edge of CK and the serial data SDI and WS (when slave) are read on the falling edge.

Control of the WS Inversion

It is possible to invert the default WS signal polarity for master and slave modes, for all protocols, by setting WSINV. By default the WS polarity is the following:

- In I2S Philips Standard, WS is LOW for left channel, and HIGH for right channel
- In MSB/LSB justified mode, WS is HIGH for left channel, and LOW for right channel
- In PCM mode, the start of frame is indicated by a rising edge of WS.

When WSINV is set, the WS polarity is inverted, then:

- In I2S Philips Standard, WS is HIGH for left channel, and LOW for right channel
- In MSB/LSB justified mode, WS is LOW for left channel, and HIGH for right channel
- In PCM mode, the start of frame is indicated by a falling edge of WS.

WSINV is located into SPI/I2S configuration register (SPI_I2SCFGR).

Control of the I/Os

The SPI/I2S block allows the settling of the WS and CK signals to their inactive state before enabling the SPI/I2S thanks to the AFCNTR bit of SPI/I2S configuration register 2 (SPI_CFG2).

This can be done by programming CKPOL and WSINV using the following sequence:

Assuming that AFCNTR is initially cleared:

- Set I2SMOD = 1, (In order to inform the hardware that the CK and WS polarity is controlled via CKPOL and WSINV).
- Set bits CKPOL and WSINV to the wanted value.
- Set AFCNTR = 1.
  Then the inactive level of CK and WS I/Os is set according to CKPOL and WSINV values, even if the SPI/I2S is not yet enabled.
- Then performs the activation sequence of the I2S/PCM

Table 558 shows the level of WS and CK signals, when the AFCNTR bit is set, and before the SPI/I2S block is enabled (i.e. inactive level). Note that the level of WS depends also on the protocol selected.
Table 558. WS and CK level before SPI/I2S is enabled when AFCNTR = 1

<table>
<thead>
<tr>
<th>WSINV</th>
<th>I2SSTD</th>
<th>WS level before SPI/I2S is enabled</th>
<th>CKPOL</th>
<th>CK level before SPI/I2S is enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>I2S Std (00)</td>
<td>→ High</td>
<td>0</td>
<td>→ Low</td>
</tr>
<tr>
<td></td>
<td>Others</td>
<td>→ Low</td>
<td>1</td>
<td>→ High</td>
</tr>
<tr>
<td>1</td>
<td>I2S Std (00)</td>
<td>→ Low</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Others</td>
<td>→ High</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The bit AFCNTR must not be set, when the SPI2S is in slave mode.

Serial data line swapping

The direction of SDI and SDO depends on the IOSWP bit of SPI/I2S configuration register 2 (SPI_CFG2), and on the slave/master mode. Table 559 gives details on this feature.

Table 559. Serial data line swapping

<table>
<thead>
<tr>
<th>Direction</th>
<th>IOSWP</th>
<th>Master mode</th>
<th>Slave mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Input line</td>
<td>Output line</td>
</tr>
<tr>
<td>RX</td>
<td>0</td>
<td>SDI</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>SDO</td>
<td>-</td>
</tr>
<tr>
<td>TX</td>
<td>0</td>
<td>-</td>
<td>SDO</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-</td>
<td>SDI</td>
</tr>
<tr>
<td>Full-duplex</td>
<td>0</td>
<td>SDI</td>
<td>SDO</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>SDO</td>
<td>SDI</td>
</tr>
</tbody>
</table>

52.9.7 Startup sequence

When the bit SPE is cleared, the user is not allowed to read and write into the SPI_RXDR and SPI_TXDR registers, but the access to other registers is allowed.

When the application wants to use the SPI/I2S block the user has to proceed as follow:

1. Ensure that the SPE is cleared, otherwise write SPE to 0.
2. Program all the configuration and control registers according to the wanted configuration. Refer to Section 52.9.16 for detailed programming examples.
3. Clear all the status flags by setting the USPC, TIFREC, OVRC and UDRC bits of SPI_IFCR register. Note that if the flag SUSP is not cleared (via SUSPC bit) the CSTART control bit has no effect.
4. Set the SPE bit, in order to activate the SPI/I2S block. When this bit is set, the serial interface is still disabled, but the DMA and interrupt services are working, allowing for example, the data transfer into the TxFIFO. The generation of MCK can also be started when SPE goes to 1.
5. Set bit CSTART, in order to activate the serial interface.

As shown in Figure 740, in I2S Philips standard master TX, the generation of the WS and CK signals starts after a resynchronization delay (SYNC_DLY2) when CSTART goes to 1.
and the TxFIFO is not empty. Note that the bit clock CK is activated 4 rising edges before the falling edge of WS in order to ensure that the external slave device can detect properly WS transition. Other standards behave similarly.

**Figure 740. Startup sequence, I2S Philips standard, master**

1. As shown in the figure, MCK can be enabled as soon as the bit SPE is set. It is generated after a synchronization delay (SYNC_DLY1). See MCK generation in Section 52.9.9: Clock generator for more information.

2. Note that the level of WS and CK signals are controlled by the SPI/I2S block during the configuration phase as soon as the AFCNTR bit is set. Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after about 3 periods of CK clock (SYNC_DLY2).

**Note:** Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after about 3 periods of CK clock (SYNC_DLY2).

In slave mode, once the bit CSTART is set, the data transfer starts when the start-of-frame condition is met:

- For I2S Philips standard, the start-of-frame condition is a falling edge of WS signal. The transmission/reception starts one bit clock later. If WSINV = 1, then the start-of-frame condition is a rising edge.
- For other protocols, the start-of-frame condition is a rising edge of WS signal. The transmission/reception starts at rising edge of WS for MSB aligned protocol. The transmission/reception starts one bit clock later for PCM protocol. If WSINV = 1, then the start-of-frame condition is a falling edge.

**Figure 741** shows an example of startup sequence in I2S Philips standard, slave mode.
2416/3119 RM0481 Rev 1

Figure 741. Startup sequence, I2S Philips standard, slave

![Diagram of startup sequence](image)

Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after 2 periods of CK clock (SYNC_DLY).

52.9.8 Stop sequence

The application can stop the I2S/PCM transfers by clearing the SPE bit. In that case the communication is stopped immediately, without waiting for the end of the current frame.

In master mode it is also possible to stop the I2S/PCM transfers at the end of the current frame. For that purpose, the user has to set the bit CSUSP, and polls the CSTART bit until it goes to 0. The CSTART bit goes to 0 when the current stereo (if an I2S mode was selected) or mono sample are completely shifted in or out. Then the SPE bit can be cleared.

The Figure 742 shows an example of stop sequence in the case of master mode. The CSUSP bit is set, during the transmission of left sample, the transfer continue until the last bit of the right sample is transferred. Then CSTART and CSUSP go back to 0, CK and WS signals go back to their inactive state, and the user can clear SPE bit.

![Diagram of stop sequence](image)

Note: In slave mode, the stop sequence is only controlled by the SPE bit.
52.9.9 Clock generator

When the I2S or PCM is configured in master mode, the user needs to program the clock generator in order to produce the Frame Synchronization (WS), the bit clock (CK) and the master clock (MCK) at the desired frequency.

If the I2S or PCM is used in slave mode, there is no need to configure the clock generator.

The frequency generated on MCK, CK and WS depends mainly on I2SDIV, ODD, CHLEN and MCKOE. The bit MCKOE indicates if a master clock need to be generated or not. The master clock has a frequency 256 or 128 times higher than the frame synchronization. This master clock is often required to provide a reference clock to external audio codecs.

Note: In master mode, there is no specific constraints on the ratio between the bus clock rate (Fpclk) and the bit clock (FCK). The bus clock frequency must be high enough in order to support the data throughput.

When the master clock is generated (MCKOE = 1), the frequency of the frame synchronization is given by the following formula in I2S mode:

\[ F_{WS} = \frac{F_{i2s\_clk}}{256 \times \{2 \times I2SDIV\} + ODD} \]

and by this formula in PCM mode:

\[ F_{WS} = \frac{F_{i2s\_clk}}{128 \times \{2 \times I2SDIV\} + ODD} \]

In addition, the frequency of the MCK (F_{MCK}) is given by the formula:

\[ F_{MCK} = \frac{F_{i2s\_clk}}{(2 \times I2SDIV) + ODD} \]

When the master clock is disabled (MCKOE = 0), the frequency of the frame synchronization is given by the following formula in I2S mode:

\[ F_{WS} = \frac{F_{i2s\_clk}}{32 \times (CHLEN + 1) \times \{2 \times I2SDIV\} + ODD} \]
And by this formula in PCM mode:

\[ F_{WS} = \frac{F_{i2s\_clk}}{16 \times (CHLEN + 1) \times (2 \times I2SDIV + ODD)} \]

Where \( F_{WS} \) is the frequency of the frame synchronization, and \( F_{i2s\_clk} \) is the frequency of the kernel clock provided to the SPI/I2S block.

**Note:**
- CHLEN and ODD can be either 0 or 1.
- \( I2SDIV \) can take any values from 0 to 255 when ODD = 0, but when ODD = 1, the value \( I2SDIV = 1 \) is not allowed.
- When \( I2SDIV = 0 \), then \((2 \times I2SDIV) + ODD\) is forced to 1.

**Note:** When \((2 \times I2SDIV) + ODD\) is odd, the duty cycle of MCK or the CK signals are not 50%. Care must be taken when odd ratio is used: it can impact margin on setup and hold time. For example if \((2 \times I2SDIV) + ODD\) = 5, then the duty cycle can be 40%.

*Table 560* provides examples of clock generator programming for I2S modes.

### MCK generation

The master clock MCK is generated when the following conditions are met:
- \( I2SMOD \) must be equal to 1,
- \( I2SCFG \) must select a master mode,
- \( MCKOE \) must be set,
- \( SPE \) must be set

*Table 560. CLKGEN programming examples for usual I2S frequencies*

<table>
<thead>
<tr>
<th>( i2s_clk ) (MHz)</th>
<th>Channel length (bits)</th>
<th>( I2SDIV )</th>
<th>ODD</th>
<th>MCK</th>
<th>Sampling rate: ( F_{WS} ) (kHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.288</td>
<td>16</td>
<td>12</td>
<td>0</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>12.288</td>
<td>32</td>
<td>6</td>
<td>0</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>12.288</td>
<td>16</td>
<td>6</td>
<td>0</td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>12.288</td>
<td>32</td>
<td>3</td>
<td>0</td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>49.152</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>No</td>
<td>48</td>
</tr>
<tr>
<td>49.152</td>
<td>32</td>
<td>8</td>
<td>0</td>
<td></td>
<td>48</td>
</tr>
<tr>
<td>49.152</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td></td>
<td>96</td>
</tr>
<tr>
<td>49.152</td>
<td>32</td>
<td>4</td>
<td>0</td>
<td></td>
<td>96</td>
</tr>
<tr>
<td>49.152</td>
<td>16</td>
<td>4</td>
<td>0</td>
<td></td>
<td>192</td>
</tr>
<tr>
<td>49.152</td>
<td>32</td>
<td>2</td>
<td>0</td>
<td></td>
<td>192</td>
</tr>
</tbody>
</table>
### 52.9.10 Internal FIFOs

The I2S interface can use a dedicated FIFO for the RX and the TX path. The samples to transmit can be written into the TxFIFO via the SPI_TXDR register. The reading of RxFIFO is performed via the SPI_RXDR register.

#### Data alignment and ordering

It is possible to select the data alignment into the SPI_RXDR and SPI_TXDR registers thanks to the DATFMT bit.

Note as well that the format of the data located into the SPI_RXDR or SPI_TXDR depends as well on the way those registers are accessed via the APB bus. **Figure 744** shows the allowed settings between APB access sizes, DATFMT and DATLEN.

**Note:** *Caution must be taken when the APB access size is 32 bits, and DATLEN = 0. For read operation the RxFIFO must contain at least two data, otherwise the read data are invalid. In the same way, for write operation, the TxFIFO must have at least two empty locations, otherwise a data can be lost.*

---

**Table 560. CLKGEN programming examples for usual I2S frequencies (continued)**

<table>
<thead>
<tr>
<th>i2s clk (MHz)</th>
<th>Channel length (bits)</th>
<th>I2SDIV</th>
<th>ODD</th>
<th>MCK</th>
<th>Sampling rate: F_WS (kHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.096</td>
<td>16 or 32</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>16</td>
</tr>
<tr>
<td>24.576</td>
<td>16 or 32</td>
<td>3</td>
<td>0</td>
<td>-</td>
<td>32</td>
</tr>
<tr>
<td>49.152</td>
<td>16 or 32</td>
<td>3</td>
<td>0</td>
<td>-</td>
<td>48</td>
</tr>
<tr>
<td>12.288</td>
<td>16 or 32</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>96</td>
</tr>
<tr>
<td>49.152</td>
<td>16 or 32</td>
<td>2</td>
<td>0</td>
<td>-</td>
<td>192</td>
</tr>
<tr>
<td>61.44</td>
<td>16 or 32</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>32</td>
</tr>
<tr>
<td>98.304</td>
<td>16 or 32</td>
<td>2</td>
<td>0</td>
<td>-</td>
<td>96</td>
</tr>
<tr>
<td>196.608</td>
<td>16 or 32</td>
<td>2</td>
<td>0</td>
<td>-</td>
<td>192</td>
</tr>
</tbody>
</table>

---

**Figure 744. Data Format**

<table>
<thead>
<tr>
<th>APB Access Size</th>
<th>DATLEN</th>
<th>SPI_RXDR, SPI_TXDR (DATFMT = 0)</th>
<th>SPI_RXDR, SPI_TXDR (DATFMT = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 bits</td>
<td>0b00</td>
<td>15 valid sample 0</td>
<td>15 valid sample 0</td>
</tr>
<tr>
<td>32 bits</td>
<td>0b00</td>
<td>16 valid sample N</td>
<td>16 valid sample N</td>
</tr>
<tr>
<td>32 bits</td>
<td>0b01</td>
<td>24 valid sample 0</td>
<td>8 valid sample 0</td>
</tr>
<tr>
<td>32 bits</td>
<td>0b10</td>
<td>31 valid sample 0</td>
<td>31 valid sample 0</td>
</tr>
</tbody>
</table>

1. In I2S mode, the sample N represents the left sample, and the sample N+1 is the right sample.

2. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10). Refer to Section 52.3: SPI implementation to check the supported data size.
It is possible to generate an interrupt or a DMA request according to a programmable FIFO threshold levels. The FIFO threshold is common to RX and TxFIFOs can be adjusted via FTHLV.

In I2S mode, the left and right audio samples are interleaved into the FIFOs. It means that for transmit operations, the user has to start to fill-up the TxFIFO with a left sample, followed by a right sample, and so on. For receive mode, the first data read from the RxFIFO is supposed to represent a left channel, the next one is a right channel, and so on.

Note that the read and write pointers of the FIFOs are reset when the bit SPE is cleared.

Refer to Section 52.9.11 and Section 52.9.15 for additional information.

**FIFO size optimization**

The basic element of the FIFO is the byte. This allows an optimization of the FIFO locations. For example when the data size is fixed to 24 bits, each audio sample takes 3 basic FIFO elements.

For example, a FIFO with 16 basic elements can have a depth of:
- 8 samples, if the DATLEN = 0 (16 bits),
- 5 samples, if the DATLEN = 1 (24 bits)(a),
- 4 samples, if the DATLEN = 2 (32 bits)(a).

### 52.9.11 FIFOs status flags

Two status flags are provided for the application to fully monitor the state of the I2S interface. Both flags can generate an interrupt request. The receive interrupt is generated if RXPIE bit is enabled, the transmit interrupt is generated if TXPIE bit is enabled. Those bits are located into the SPI_IER register.

**TxFIFO threshold reached (TXP)**

When set, this flag indicates that the TxFIFO contains at least FTHLV empty locations. thus FTHLV new data to be transmitted can be written into SPI_TXDR. The TXP flag is reset when the amount of empty locations is lower than FTHLV. Note that TXP = 1, when the I2S is disabled (SPE bit is reset).

**RxFIFO threshold reached (RXP)**

When set, this flag indicates that there is at least FTHLV valid data into the RxFIFO, thus the user can read those data via SPI_RXDR. It is reset when the RxFIFO contains less than FTHLV data.

See Section 52.10 for additional information on interrupt function in I2S mode.

### 52.9.12 Handling of underrun situation

In transmit mode, an underrun situation is detected when a new data needs to be loaded into the shift register while the TxFIFO is already empty. In such a situation the UDR flag is set, and at least one audio frame contains unexpected data.

---

(a) Not always available, refer to Section 52.3: SPI implementation in order to check if 24 and 32-bit data width are supported.
For I2S modes, there is a hardware mechanism to prevent misalignment situations (left and right channel swapped). When an underrun situation is detected, the last valid data present in the shift register is repeated if the real channel length matches the length selected by the CHLEN bit. In the other cases, an undefined data is repeated. Typically, if the block is programmed in slave TX mode, and the external master audio device is using channel lengths other than 16 or 32 bits, the repeated data value is undefined in case of underrun.

The following figure shows the case where an underrun occurs and the peripheral re-plays the last valid data on left and right channels as long as conditions of restart are not met. The transmission restarts:

- When there is enough data into the TxFIFO, and
- When the UDR flag is cleared by the software, and
- When the restart condition is met:
  - if the underrun occurs when a right channel data needs to be transmitted, the transmission restarts when a right channel needs to be transmitted, or
  - if the underrun occurs when a left channel data needs to be transmitted, the transmission restarts when a left channel needs to be transmitted.

![Figure 745. Handling of underrun situation](image)

When the block is configured in one of the PCM modes, the transmission restarts at the start of the next frame, when there is enough data in the TxFIFO, and the UDR flag is cleared.

The UDR flag can trigger an interrupt if the UDRIE bit in the SPI_IER register is set. The UDR bit is cleared by writing UDRC bit of SPI_IFCR register to 1.

**Note:**

An underrun situation can occur in master or slave mode. In master mode, when an underrun occurs, the WS, CK and MCK signal are not gated.

Due to resynchronization, any change on the UDR flag is taken into account by the hardware after at least 2 periods of CK clock.

### 52.9.13 Handling of overrun situation

In receive mode, an overrun situation is detected when the shift register needs to store a new data into the RxFIFO, while the RxFIFO is full. In such a situation the OVR flag is set, and the incoming data is lost.
In I2S mode, there is a hardware mechanism in to prevent misalignment situations (left and right channel swapped). As shown in the following figure, when an overrun occurs, the peripheral stops writing data into the Rx_FIFO as long as restart conditions are not met.

The reception restarts when there is enough room into the Rx_FIFO, and the OVR flag is cleared, the block starts by writing next the right channel into the Rx_FIFO if the overrun occurs when a right channel data is received or by writing the next left channel if the overrun occurs when a left channel data is received.

![Figure 746. Handling of overrun situation](image)

When the block is configured in PCM mode, after an overrun error, the block stops writing data in the Rx_FIFO as long as conditions of restart are not met. When there is enough room in the Rx_FIFO, and the OVR flag is cleared, the next received data are written in the Rx_FIFO.

An interrupt can be generated if the OVRIE bit of the SPI_IER register is set. The OVR bit is cleared by writing OVRC bit of SPI_IFCR register to 1.

**Note:** An overrun situation can occur in master or slave mode. In master mode when an overrun occurs, the WS, CK and MCK signal are not gated.

### 52.9.14 Frame error detection

When configured in slave mode, the SPI/I2S block detects two kinds of frame errors:
- A frame synchronization received while the shift-in or shift-out of the previous data is not completed (early frame error). This mode is selected with FIXCH = 0.
- A frame synchronization occurring at an unexpected position. This mode is selected with FIXCH = 1.

In slave mode, if the frame length provided by the external master device is different from 32 or 64 bits, the user has to clear FIXCH. As the SPI/I2S synchronize each transfer with the WS there is no misalignment risk, but in a noisy environment, if a glitch occurs in the CK signal, a sample may be affected and the application is not aware of this.

If the frame length provided by the external master device is equal to 32 or 64 bits, then the user can set FIXCH and adjust accordingly CHLEN. As the SPI/I2S synchronize each transfer with the WS there is still no misalignment risk, and if the amount of bit clock between each channel boundary is different from CHLEN, the frame error flag (TIFRE) is set.
Figure 747 shows an example of frame error detection. The SPI/I2S block is in slave mode and the amount of bit clock periods for left channel are not enough to shift-in or shift-out the data. The figure shows that the on-going transfer is interrupted and the next one is started in order to remain aligned to the WS signal.

Figure 747. Frame error detection, with FIXCH=0

An interrupt can be generated if the TIFREIE bit is set. The frame error flag (TIFRE) is cleared by writing the TIFREC bit of the SPI_IFCR register to 1.

It is possible to extend the coverage of the frame error flag by setting the bit FIXCH. When this bit is set, then the SPI/I2S is expecting fixed channel lengths in slave mode. This means that the expected channel length can be 16 or 32 bits, according to CHLEN. As shown in Figure 748, in this mode the SPI/I2S block is able to detect if the WS signal is changing at the expected moment (too early or too late).

Note: Figure 747 and Figure 748 show the mechanism for the slave transmit mode, but this is also true for slave receive and slave full-duplex.

Figure 748. Frame error detection, with FIXCH=1

The frame error detection can be generally due to noisy environment disturbing the good reception of WS or CK signals.

Note: The SPI/I2S is not able to recover properly if an overrun and an early frame occur within the same frame. In this case the user has to disable and re-enable the SPI/I2S.
52.9.15 DMA interface

The I2S/PCM mode shares the same DMA requests lines than the SPI function. There is a separated DMA channel for TX and RX paths. Each DMA channel can be enabled via RXDMAEN and TXDMAEN bits of SPI_CFG1 register.

In receive mode, the DMA interface is working as follow:
1. The hardware evaluates the RxFIFO level,
2. If the RxFIFO contains at least FTHLV samples, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the RxFIFO contains less than FTHLV samples, no DMA request is generated, and the hardware loop to step 1

In transmit mode, the DMA interface is working as follow:
1. The hardware evaluates the TxFIFO level,
2. If the TxFIFO contains at least FTHLV empty locations, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the TxFIFO contains less than FTHLV empty locations, no DMA request is generated, and the hardware loop to step 1

52.9.16 Programming examples

Master I2S Philips standard, full-duplex

This example shows how to program the interface for supporting the I2S Philips standard protocol in master full-duplex mode, with a sampling rate of 48 kHz, using the master clock. The assumption has been taken that the SPI/I2S is receiving a kernel clock (i2s_clk) of 61.44 MHz from the clock controller of the circuit. In the example above we took the assumption that the external audio codec needs to be programmed, for example via an I2C interface before starting the transfer. In addition, it is supposed that this external audio codec needs the MCK to accept I2C commands.
**Procedure**

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed.
2. Program the AFMUX in order to select the wanted I/Os. In the current example MCK, CK, WS, SDO, SDI are needed.
3. Program the clock generator to provide the MCK clock, and to have a frame synchronization rate at exactly 48 kHz. Set i2SDIV to 2, ODD to 1, and MCKOE to 1.
4. Program the serial interface protocol: CKPOL = 0, WSINV = 0, CHLEN = 1 (32 bits per channel) DATLEN = 1 (24 bits), I2SSTD = 0 (I2S Philips standard), I2SCFG = 5 (master Full-duplex), I2SMOD = 1, for I2S/PCM mode.
5. Adjust the FIFO threshold, by setting the wanted value into FTHLV. For example, if a threshold of 2 audio samples is required, FTHLV = 1.
6. If the application wishes to perform data transfer via DMA, set the bits TXDMAEN and RXDMAEN.
7. Clear all interrupt enable fields located in SPI_IER register.
8. Clear all status flags, by setting SUSPC, TIFREC, OVRC, UDRC of SPI_IFCR register.
9. Set SPE bit, then the MCK is generated. In the example presented here, the TxFIFO is not filled-up as soon as SPE = 1. The application can then configure the external audio codec via an I2C interface even if audio samples to transmit are not yet available. An **Alternative sequence** is proposed hereafter.
10. When the application wants to start the data transfer:
   - If the data transfer uses DMA:
     - Program the DMA peripheral: two channels, one for RX and one for TX
     - Initialize the memory buffer with valid audio samples for TX path – Enable the DMA channels,
     - Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
     - Enable the DMA channel. If TXDMAEN was set, the TxFIFO is filled-up.
   - If the data transfer is done via interrupt:
     - Enable the interrupt events UDRIE, OVRIE, TXPIE and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.
11. Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into SPI_CR1 register.

**Alternative sequence**

- Steps 1 to 8 similar to the previous sequence.
- If the data transfer uses DMA:
  - Program the DMA peripheral: two channels, one for RX and one for TX
  - Initialize the memory buffer with valid audio samples for TX path
  - Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
  - Enable the DMA channels,
- If the data transfer is done via interrupt:
Enable the interrupt events UDRIE, OVRIE, TXPIE and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.

- Set SPE, as soon as this bit is set to one the following actions may happen:
  - If the interrupt generation is enabled, the SPI/I2S generates an interrupt request allowing the interrupt handler to fill-up the TxFIFO.
  - If the DMA transfer are enabled, the SPI/I2S generates DMA requests in order to fill-up the TxFIFO
  - The MCK is generated. The application can then configure the external audio codec via an I2C interface.

- Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into SPI_CR1 register.

Stop Procedure in master mode

1. Set the bit CSUSP, in order to stop on-going transfers
2. Check the value of CSTART bit until it goes to 0
3. Clear the SUSP flag by setting SUSPC
4. Stop DMA peripheral, bus clock...
5. Clear bit SPE in order to disable the SPI/I2S block

Slave I2S Philips standard, receive

This example shows how to program the interface for supporting the I2S Philips standard protocol in slave receiver mode, with a sampling rate of 48 kHz. Note that in slave mode the SPI/I2S block cannot control the sample rate of the received samples. In this example we took the assumption that the external master device is delivering an I2S frame structure with a channel length of 24 bits. So we cannot use the capability offered for frame error detection when FIXCH is set.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed,
2. Program the AFMUX in order to select the wanted I/Os. In the current example CK, WS, SDI,
3. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, FIXCH = 0 (because channel length is different from 16 and 32 bits), DATLEN = 0 (16 bits), I2SSTD = 0 (Philips protocol), I2SCFG = 1 (slave RX), I2SMOD = 1, for I2S mode.
4. Adjust the FIFO threshold, by setting the wanted value into FTHLV. For example if a threshold of 2 audio samples is required, FTHLV = 1.
5. Clear all status flag registers.
6. Enable the flags that generate an interrupt such as OVRIE and TIFRE.
7. If the data transfer uses DMA:
   - Program the DMA peripheral: one RX channel
   - Enable the DMA channel,
   - In the SPI/I2S block, enable the DMA by setting the RXDMAEN bit.
8. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the RXPIE bit.
9. Set SPE.
10. Finally the user can set the bit CSTART in order to enable the serial interface. The SPI/I2S starts to store data into the RxFIFO on the next occurrence of left data transmitted by the external master device.

Stop Procedure in slave mode
1. Clear bit SPE in order to disable the SPI/I2S block
2. Stop DMA peripheral, bus clock...

52.10 I2S interrupts

In PCM/I2S mode an interrupt (spi_it) or a wake-up event signal (spi_wkup) can be generated according to the events described in the Table 561.

Interrupt events can be enabled and disabled separately.

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Event/Interrupt clearing method</th>
<th>Exit Sleep mode</th>
<th>Exit Stop modes</th>
<th>Exit Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI</td>
<td>TxFIFO threshold reached</td>
<td>TXP</td>
<td>When the TxFIFO contains less than FTHLV empty locations</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Rx FIFO threshold reached</td>
<td>RXP</td>
<td>When the Rx FIFO contains less than FTHLV samples</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overrun error</td>
<td>OVR</td>
<td>Write OVRC to 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Underrun error</td>
<td>UDR</td>
<td>Write UDRC to 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Frame error flag</td>
<td>TIFRE</td>
<td>Write TIFREC to 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

52.11 SPI/I2S registers

52.11.1 SPI/I2S control register 1 (SPI_CR1)

Address offset: 0x00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rs</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

RM0481 Rev 1 2427/3119
Bits 31:17  Reserved, must be kept at reset value.

Bit 16  **IOLOCK**: locking the AF configuration of associated I/Os
   This bit can be changed by software only when SPI is disabled (SPE = 0). It is cleared by
   hardware if a MODF event occurs
   0: AF configuration is not locked
   1: AF configuration is locked
   When this bit is set, SPI_CFG2 register content cannot be modified. This bit is write-
   protected when SPI is enabled (SPE = 1).

Bit 15  **TCRCINI**: CRC calculation initialization pattern control for transmitter
   0: all zero pattern is applied
   1: all ones pattern is applied

Bit 14  **RCRCINI**: CRC calculation initialization pattern control for receiver
   0: All zero pattern is applied
   1: All ones pattern is applied

Bit 13  **CRC33_17**: 32-bit CRC polynomial configuration
   0: Full size (33-bit or 17-bit) CRC polynomial is not used
   1: Full size (33-bit or 17-bit) CRC polynomial is used

Bit 12  **SSI**: internal SS signal input level
   This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the
   peripheral SS input internally and the I/O value of the SS pin is ignored.

Bit 11  **HDDIR**: Rx/Tx direction at Half-duplex mode
   In Half-Duplex configuration the HDDIR bit establishes the Rx/Tx direction of the data
   transfer. This bit is ignored in Full-Duplex or any Simplex configuration.
   0: SPI is receiver
   1: SPI is transmitter

Bit 10  **CSUSP**: master suspend request
   This bit reads as zero.
   In Master mode, when this bit is set by software, the CSTART bit is reset at the end of the
   current frame and communication is suspended. The user has to check SUSP flag to check
   end of the frame transaction.
   The Master mode communication must be suspended (using this bit or keeping TXDR
   empty) before going to Low-power mode.
   After software suspension, SUSP flag must be cleared and SPI disabled and re-enabled
   before the next transaction starts.

Bit 9  **CSTART**: master transfer start
   This bit can be set by software if SPI is enabled only to start an SPI or I2S/PCM
   communication. In SPI mode, it is cleared by hardware when end of transfer (EOT) flag is set
   or when a transaction suspend request is accepted. In I2S/PCM mode, it is also cleared by
   hardware as described in the Section 52.9.8: Stop sequence.
   0: master transfer is at idle
   1: master transfer is ongoing or temporary suspended by automatic suspend
   In SPI mode, the bit is taken into account at master mode only. If transmission is enabled,
   communication starts or continues only if any data is available in the transmission FIFO.
Bit 8 **MASRX**: master automatic suspension in Receive mode

This bit is set and cleared by software to control continuous SPI transfer in master receiver mode and automatic management in order to avoid overrun condition.

0: SPI flow/clock generation is continuous, regardless of overrun condition. (data are lost)
1: SPI flow is suspended temporarily on RxFIFO full condition, before reaching overrun condition. The SUSP flag is set when the SPI communication is suspended.

When SPI communication is suspended by hardware automatically, it may happen that few bits of next frame are already clocked out due to internal synchronization delay.

This is why, the automatic suspension is not quite reliable when size of data drops below 8 bits. In this case, a safe suspension can be achieved by combination with delay inserted between data frames applied when MIDI parameter keeps a non zero value; sum of data size and the interleaved SPI cycles should always produce interval at length of 8 SPI clock periods at minimum. After software clearing of the SUSP bit, the communication resumes and continues by subsequent bits transaction without any next constraint. Prior the SUSP bit is cleared, the user must release the RxFIFO space as much as possible by reading out all the data packets available at RxFIFO based on the RXP flag indication to prevent any subsequent suspension.

**Bits 7:1** Reserved, must be kept at reset value.

Bit 0 **SPE**: serial peripheral enable

This bit is set by and cleared by software.

0: Serial peripheral disabled.
1: Serial peripheral enabled

When SPE = 1, SPI data transfer is enabled, SPI_CFG1 and SPI_CFG2 configuration registers, CRCPOLY, UDRDR, IOLOCK bit in the SPI_CR1 register are write protected. They can be changed only when SPE = 0.

When SPE = 0 any SPI operation is stopped and disabled, all the pending requests of the events with enabled interrupt are blocked except the MODF interrupt request (but their pending still propagates the request of the spi_plck clock), the SS output is deactivated at master, the RDY signal keeps not ready status at slave, the internal state machine is reseted, all the FIFOs content is flushed, CRC calculation initialized, receive data register is read zero. SPE is cleared and cannot be set when MODF error flag is active.

### 52.11.2 SPI/I2S control register 2 (SPI_CR2)

Address offset: 0x04

Reset value: 0x0000_0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**TSIZE[15:0]**

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

**STI**
52.11.3 SPI/I2S configuration register 1 (SPI_CFG1)

Address offset: 0x08
Reset value: 0x0007 0007

The content of this register is write protected when SPI is enabled.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
</tr>
</tbody>
</table>

Bit 31 **BPASS**: bypass of the prescaler at master baud rate clock generator
- 0: bypass is disabled
- 1: bypass is enabled

Bits 30:28 **MBR[2:0]**: master baud rate prescaler setting
- 000: SPI master clock/2
- 001: SPI master clock/4
- 010: SPI master clock/8
- 011: SPI master clock/16
- 100: SPI master clock/32
- 101: SPI master clock/64
- 110: SPI master clock/128
- 111: SPI master clock/256

*Note: MBR setting is considered at slave working at TI mode, too (see Section 52.5.1: TI mode).*

Bits 27:23 Reserved, must be kept at reset value.

Bit 22 **CRCEN**: hardware CRC computation enable
- 0: CRC calculation disabled
- 1: CRC calculation enabled

Bit 21 Reserved, must be kept at reset value.
Bits 20:16 **CRCSIZE[4:0]**: length of CRC frame to be transacted and compared

Most significant bits are taken into account from polynomial calculation when CRC result is transacted or compared. The length of the polynomial is not affected by this setting.

- 00000: reserved
- 00001: reserved
- 00010: reserved
- 00011: 4-bits
- 00100: 5-bits
- 00101: 6-bits
- 00110: 7-bits
- 00111: 8-bits
- .....  
- 11101: 30-bits
- 11110: 31-bits
- 11111: 32-bits

The value must be set equal or multiply of data size (DSIZE[4:0]). Its maximum size corresponds to DSIZE maximum at the instance.

*Note: The most significant bit at CRCSIZE bit field is reserved at the peripheral instances where data size is limited to 16-bit.*

Bit 15 **TXDMAEN**: Tx DMA stream enable

- 0: Tx DMA disabled
- 1: Tx DMA enabled

Bit 14 **RXDMAEN**: Rx DMA stream enable

- 0: Rx-DMA disabled
- 1: Rx-DMA enabled

Bits 13:10 Reserved, must be kept at reset value.
Bit 9 **UDRCFG**: behavior of slave transmitter at underrun condition

0: slave sends a constant pattern defined by the user at the SPI_UDRDR register
1: Slave repeats lastly received data from master. When slave is configured at transmit only mode (COMM[1:0] = 01), all zeros pattern is repeated.

For more details see Figure 729: Optional configurations of slave detecting underrun condition.

Bits 8:5 **FTHLV[3:0]**: FIFO threshold level

Defines number of data frames at single data packet. Size of the packet should not exceed 1/2 of FIFO space.

- 0000: 1-data
- 0001: 2-data
- 0010: 3-data
- 0011: 4-data
- 0100: 5-data
- 0101: 6-data
- 0110: 7-data
- 0111: 8-data
- 1000: 9-data
- 1001: 10-data
- 1010: 11-data
- 1011: 12-data
- 1100: 13-data
- 1101: 14-data
- 1110: 15-data
- 1111: 16-data

SPI interface is more efficient if configured packet sizes are aligned with data register access parallelism:

- If SPI data register is accessed as a 16-bit register and DSIZE ≤ 8 bit, better to select FTHLV = 2, 4, 6.
- If SPI data register is accessed as a 32-bit register and DSIZE > 8 bit, better to select FTHLV = 2, 4, 6, while if DSIZE ≤ 8bit, better to select FTHLV = 4, 8, 12.

*Note: FTHLV[3:2] bits are reserved at instances with limited set of features*
Bits 4:0 **DSIZE[4:0]**: number of bits in a single SPI data frame
- 00000: not used
- 00001: not used
- 00010: not used
- 00011: 4 bits
- 00100: 5 bits
- 00101: 6 bits
- 00110: 7 bits
- 00111: 8 bits
- ....
- 11101: 30 bits
- 11110: 31 bits
- 11111: 32 bits

*Note:* Maximum data size can be limited up to 16-bits at some instances. At instances with limited set of features, **DSIZE[2:0]** bits are reserved and must be kept at reset state. **DSIZE[4:3]** bits then control next settings of data size:
- 00xxx: 8-bits
- 01xxx: 16-bits
- 10xxx: 24-bits
- 11xxx: 32-bits.

### 52.11.4 SPI/I2S configuration register 2 (SPI_CFG2)

Address offset: 0x0C

Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled or IOLOCK bit is set at SPI_CR1 register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

STI
Bit 31 **AFCNTR**: alternate function GPIOs control
This bit is taken into account when SPE = 0 only
0: The peripheral takes no control of GPIOs while it is disabled
1: The peripheral keeps always control of all associated GPIOs
When SPI must be disabled temporarily for a specific configuration reason (for example CRC reset, CPHA or HDDIR change) setting this bit prevents any glitches on the associated outputs configured at alternate function mode by keeping them forced at state corresponding to the current SPI configuration.

*Note*: This bit can be also used in PCM and I2S modes.

*Note*: The bit AFCNTR must not be set, when the block is in slave mode.

Bit 30 **SSOM**: SS output management in Master mode
This bit is taken into account in Master mode when SSOE is enabled. It allows the SS output to be configured between two consecutive data transfers.
0: SS is kept at active level till data transfer is completed, it becomes inactive with EOT flag
1: SPI data frames are interleaved with SS non active pulses when MIDI[3:0]>1

Bit 29 **SSOE**: SS output enable
This bit is taken into account in Master mode only
0: SS output is disabled and the SPI can work in multimaster configuration
1: SS output is enabled. The SPI cannot work in a multimaster environment. It forces the SS pin at inactive level after the transfer is completed or SPI is disabled with respect to SSOM, MIDI, MSSI, SSIOP bits setting

Bit 28 **SSIOP**: SS input/output polarity
0: low level is active for SS signal
1: high level is active for SS signal

Bit 27 Reserved, must be kept at reset value.

Bit 26 **SSM**: software management of SS signal input
0: SS input value is determined by the SS PAD
1: SS input value is determined by the SSI bit
When master uses hardware SS output (SSM = 0 and SSOE = 1) the SS signal input is forced to not active state internally to prevent master mode fault error.

Bit 25 **CPOL**: clock polarity
0: SCK signal is at 0 when idle
1: SCK signal is at 1 when idle

Bit 24 **CPHA**: clock phase
0: the first clock transition is the first data capture edge
1: the second clock transition is the first data capture edge

Bit 23 **LSBFRST**: data frame format
0: MSB transmitted first
1: LSB transmitted first

*Note*: This bit can be also used in PCM and I2S modes.

Bit 22 **MASTER**: SPI Master
0: SPI Slave
1: SPI Master
Bits 21:19 **SP[2:0]**: serial protocol
- 000: SPI Motorola
- 001: SPI TI
- others: reserved, must not be used

Bits 18:17 **COMM[1:0]**: SPI Communication Mode
- 00: full-duplex
- 01: simplex transmitter
- 10: simplex receiver
- 11: half-duplex

Bit 16 Reserved, must be kept at reset value.

Bit 15 **IOSWP**: swap functionality of MISO and MOSI pins
- 0: no swap
- 1: MOSI and MISO are swapped
  When this bit is set, the function of MISO and MOSI pins alternate functions are inverted.
  Original MISO pin becomes MOSI and original MOSI pin becomes MISO.
  *Note: This bit can be also used in PCM and I2S modes to swap SDO and SDI pins.*

Bit 14 **RDIOP**: RDY signal input/output polarity
- 0: high level of the signal means the slave is ready for communication
- 1: low level of the signal means the slave is ready for communication

Bit 13 **RDIOM**: RDY signal input/output management
- 0: RDY signal is defined internally fixed as permanently active (RDIOP setting has no effect)
- 1: RDY signal is overtaken from alternate function input (at master case) or output (at slave case) of the dedicated pin (RDIOP setting takes effect)

  *Note: When DSIZE at the SPI_CFG1 register is configured shorter than 8-bit, the RDIOM bit must be kept at zero.*

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:4 **MIDI[3:0]**: master Inter-Data Idleness
  Specifies minimum time delay (expressed in SPI clock cycles periods) inserted between two consecutive data frames in Master mode.
- 0000: no delay
- 0001: 1 clock cycle period delay
- ... 1111: 15 clock cycle periods delay

  *Note: This feature is not supported in TI mode.*

Bits 3:0 **MSSI[3:0]**: Master SS Idleness
  Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of SS opening a session and the beginning of the first data frame of the session in Master mode when SSOE is enabled.
- 0000: no extra delay
- 0001: 1 clock cycle period delay added
- ... 1111: 15 clock cycle periods delay added

  *Note: This feature is not supported in TI mode.*
  *To include the delay, the SPI must be disabled and re-enabled between sessions.*
### 52.11.5 SPI/I2S interrupt enable register (SPI_IER)

Address offset: 0x10  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:10</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>
| 9   | MODFIE: mode Fault interrupt enable | 0: MODF interrupt disabled  
0: MODF interrupt enabled |
| 8   | TIFREIE: TIFRE interrupt enable | 0: TIFRE interrupt disabled  
0: TIFRE interrupt enabled |
| 7   | CRCEIE: CRC error interrupt enable | 0: CRC interrupt disabled  
0: CRC interrupt enabled |
| 6   | OVRIE: OVR interrupt enable | 0: OVR interrupt disabled  
0: OVR interrupt enabled |
| 5   | UDRIE: UDR interrupt enable | 0: UDR interrupt disabled  
0: UDR interrupt enabled |
| 4   | TXTFIE: TXTFIE interrupt enable | 0: TXTF interrupt disabled  
0: TXTF interrupt enabled |
| 3   | EOTIE: EOT, SUSP and TXC interrupt enable | 0: EOT/SUSP/TXC interrupt disabled  
0: EOT/SUSP/TXC interrupt enabled |
| 2   | DXPIE: DXP interrupt enabled | 0: DXP interrupt disabled  
0: DXP interrupt enabled |
| 1   | TXPIE: TXP interrupt enable | 0: TXP interrupt disabled  
0: TXP interrupt enabled |
| 0   | RXPIE: RXP interrupt enable | 0: RXP interrupt disabled  
0: RXP interrupt enabled |
52.11.6 SPI/I2S status register (SPI_SR)

Address offset: 0x14

Reset value: 0x0000 1002

All the flags of this register are not cleared automatically when the SPI is re-enabled. They require specific clearing access exclusively via the flag clearing register as noted in the bits descriptions below.

<table>
<thead>
<tr>
<th>Bit 31:16 CTSIZE[15:0]</th>
<th>number of data frames remaining in current TSIZE session</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r r r r</td>
<td>The value is not quite reliable when traffic is ongoing on bus .</td>
</tr>
<tr>
<td>Note: CTSIZE[15:0] bits are not available in instances with limited set of features.</td>
<td></td>
</tr>
</tbody>
</table>

Bit 15 RXWNE: RxFIFO word not empty
0: less than four bytes of RxFIFO space is occupied by data
1: at least four bytes of RxFIFO space is occupied by data

Note: This bit value does not depend on DSIZE setting and keeps together with RXPLVL[1:0] information about RxFIFO occupancy by residual data.

Bits 14:13 RXPLVL[1:0]: RxFIFO packing level
When RXWNE = 0 and data size is set up to 16-bit, the value gives number of remaining data frames persisting at RxFIFO.
00: no next frame is available at RxFIFO
01: 1 frame is available
10: 2 frames are available*
11: 3 frames are available*

Note: (*) Possible value when data size is set up to 8-bit only.
When data size is greater than 16-bit, these bits are always read as 00. In that consequence, the single data frame received at the FIFO cannot be detected neither by RXNE nor by RXPLVL bits if data size is set from 17 to 24 bits. The user must then apply other methods to detect the number of data received, such as monitor the EOT event when TSIZE > 0 or RXP events when FTHLV = 0.

Bit 12 TXC: TxFIFO transmission complete
The flag behavior depends on TSIZE setting.
When TSIZE = 0, the TXC is changed by hardware exclusively and it raises each time the TxFIFO becomes empty and there is no activity on the bus.
If TSIZE ≠ 0 there is no specific reason to monitor TXC as it just copies the EOT flag value including its software clearing. The TXC generates an interrupt when EOTIE is set.
This flag is set when SPI is reset or disabled.
0: current data transaction is still ongoing, data is available in TxFIFO or last frame transmission is on going.
1: last TxFIFO frame transmission complete
Bit 11 **SUSP**: suspension status

In Master mode, SUSP is set by hardware either as soon as the current frame is completed after CSUSP request is done or at master automatic suspend receive mode (MASRX bit is set at SPI_CR1 register) on RxFIFO full condition.

SUSP generates an interrupt when EOTIE is set.

This bit must be cleared prior SPI is disabled and this is done by writing 1 to SUSPC bit of SPI_IFCR exclusively.

- **0**: SPI not suspended (Master mode active or other mode).
- **1**: Master mode is suspended (current frame completed).

Bit 10 **Reserved**, must be kept at reset value.

Bit 9 **MODF**: mode fault

- **0**: no mode fault
- **1**: mode fault detected.

When MODF is set, SPE and IOLOCK bits of SPI_CR1 register are reset and setting SPE again is blocked until MODF is cleared.

This bit is cleared by writing 1 to MODFC bit of SPI_IFCR exclusively.

Bit 8 **TIFRE**: TI frame format error

- **0**: no TI Frame Error
- **1**: TI frame error detected

This bit is cleared by writing 1 to TIFREC bit of SPI_IFCR exclusively.

Bit 7 **CRCE**: CRC error

- **0**: no CRC error
- **1**: CRC error detected

This bit is cleared when SPI is re-enabled or by writing 1 to CRCEC bit of SPI_IFCR optionally.

Bit 6 **OVR**: overrun

- **0**: no overrun
- **1**: overrun detected

This bit is cleared when SPI is re-enabled or by writing 1 to OVRC bit of SPI_IFCR optionally.

Bit 5 **UDR**: underrun

- **0**: no underrun
- **1**: underrun detected

This bit is cleared when SPI is re-enabled or by writing 1 to UDRC bit of SPI_IFCR optionally.

**Note**: In SPI mode, the UDR flag applies to Slave mode only. In I2S/PCM mode, (when available) this flag applies to Master and Slave mode.

Bit 4 **TXTF**: transmission transfer filled

- **0**: upload of TxFIFO is ongoing or not started
- **1**: TxFIFO upload is finished

TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the TxFIFO.

This bit is cleared by software write 1 to TXTFC bit of SPI_IFCR exclusively.

TXTF flag triggers an interrupt if TXTFIE bit is set.

TXTF setting clears the TXPIE and DXPIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.
Bit 3 EOT: end of transfer
EOT is set by hardware as soon as a full transfer is complete, that is when SPI is re-enabled or when TSIZE number of data have been transmitted and/or received on the SPI. EOT is cleared when SPI is re-enabled or by writing 1 to EOTC bit of SPI_IFCR optionally. EOT flag triggers an interrupt if EOTIE bit is set.
If DXF flag is used until TXTF flag is set and DXPIE is cleared, EOT can be used to download the last packets contained into Rx FIFO in one-shot.
0: transfer is ongoing or not started
1: transfer complete
In master, EOT event terminates the data transaction and handles SS output optionally. When CRC is applied, the EOT event is extended over the CRC frame transaction.
To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transaction starts despite its setting is not changed.

Bit 2 DXP: duplex packet
0: TxFIFO is Full and/or RxFIFO is Empty
1: both TxFIFO has space for write and RxFIFO contains for read a single packet at least
DXP flag is set whenever both TXP and RXP flags are set regardless SPI mode.

Bit 1 TXP: Tx-packet space available
0: not enough free space at TxFIFO to host next data packet
1: enough free space at TxFIFO to host at least one data packet
TXP flag can be changed only by hardware. Its value depends on the physical size of the FIFO and its threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and respective DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is stored by performing consecutive write operations to SPI_TXDR, TXP flag must be checked again once a complete data packet is stored at TxFIFO. TXP is set despite SPI TxFIFO becomes inaccessible when SPI is reset or disabled.

Bit 0 RXP: Rx-packet available
0: RxFIFO is empty or an incomplete data packet is received
1: RxFIFO contains at least one data packet
The flag is changed by hardware. It monitors the total number of data currently available at RxFIFO if SPI is enabled. RXP value depends on the FIFO threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is read by performing consecutive read operations from SPI_RXDR, RXP flag must be checked again once a complete data packet is read out from RxFIFO.

52.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR)
Address offset: 0x18
Reset value: 0x0000 0000
Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SUSPC**: Suspend flag clear
Writing a 1 into this bit clears SUSP flag in the SPI_SR register

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODFC**: mode fault flag clear
Writing a 1 into this bit clears MODF flag in the SPI_SR register

Bit 8 **TIFREC**: Ti frame format error flag clear
Writing a 1 into this bit clears TIFRE flag in the SPI_SR register

Bit 7 **CRCCEC**: CRC error flag clear
Writing a 1 into this bit clears CRCE flag in the SPI_SR register

Bit 6 **OVRC**: overrun flag clear
Writing a 1 into this bit clears OVR flag in the SPI_SR register

Bit 5 **UDRC**: underrun flag clear
Writing a 1 into this bit clears UDR flag in the SPI_SR register

Bit 4 **TXTFC**: transmission transfer filled flag clear
Writing a 1 into this bit clears TXTF flag in the SPI_SR register

Bit 3 **EOTC**: end of transfer flag clear
Writing a 1 into this bit clears EOT flag in the SPI_SR register

Bits 2:0 Reserved, must be kept at reset value.

**52.11.8 SPI/I2S transmit data register (SPI_TXDR)**

Address offset: 0x20

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

**TXDR[31:16]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

**TXDR[15:0]**

Bits 31:0 **TXDR[31:0]**: transmit data register

The register serves as an interface with TxFIFO. A write to it accesses TxFIFO.

**Note:** In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are ignored when writing to the register, and read as zero when the register is read.

**Note:** DR can be accessed byte-wise (8-bit access): in this case only one data-byte is written by single access.

- halfword-wise (16 bit access) in this case 2 data-bytes or 1 halfword-data can be written by single access.
- word-wise (32 bit access). In this case 4 data-bytes or 2 halfword-data or word-data can be written by single access.

Write access of this register less than the configured data size is forbidden.
52.11.9  **SPI/I2S receive data register (SPI_RXDR)**  
Address offset: 0x30  
Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
</tr>
</tbody>
</table>

**RXDR[31:16]**

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

**Bits 31:0 RXDR[31:0]: receive data register**

The register serves as an interface with RxFIFO. When it is read, RxFIFO is accessed.

*Note: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are read as zero when the register is read.*

*Writing to the register is ignored.*

*Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is read by single access*

*halfword-wise (16 bit access) in this case 2 data-bytes or 1 halfword-data can be read by single access*

*word-wise (32 bit access). In this case 4 data-bytes or 2 halfword-data or word-data can be read by single access.*

*Read access of this register less than the configured data size is forbidden.*

52.11.10  **SPI/I2S polynomial register (SPI_CRCPOLY)**

Address offset: 0x40  
Reset value: 0x0000 0107

The content of this register is write protected when SPI is enabled.

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
</tr>
</tbody>
</table>

**CRCPOLY[31:16]**

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

**CRCPOLY[15:0]**
Bits 31:0 **CRCPOLY[31:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The default 9-bit polynomial setting 0x107 corresponds to default 8-bit setting of DSIZE. It is compatible with setting 0x07 used in other ST products with fixed length of the polynomial string, where the most significant bit of the string is always kept hidden.

Length of the polynomial is given by the most significant bit of the value stored in this register. It must be set greater than DSIZE. CRC33_17 bit must be set additionally with CRCPOLY register when DSIZE is configured to maximum 32-bit or 16-bit size and CRC is enabled (to keep polynomial length greater than data size).

*Note:* CRCPOLY[31:16] bits are reserved at instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

### 52.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC)

*Address offset: 0x44*

*Reset value: 0x0000 0000*

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**TXCRC[31:16]**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**TXCRC[15:0]**

Bits 31:0 **TXCRC[31:0]:** CRC register for transmitter

When CRC calculation is enabled, the TXCRC[31:0] bits contain the computed CRC value of the subsequently transmitted bytes. CRC calculation is initialized when the CRCEN bit of SPI_CR1 is set or when a data block is transacted completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on SPI_CRCPOLY register and CRCSIZE bits settings at SPI_CFG1 register.

*Note:* A read to this register when the communication is ongoing may return an incorrect value.

*Note:* This bitfield is not used in I2S mode.

*Note:* TXCRC[31-16] bits are reserved at instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

*Note:* The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.
52.11.12  SPI/I2S receiver CRC register (SPI_RXCRC)

Address offset: 0x48
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16 RXCRC[31:16]</th>
<th>Bits 15:0 RXCRC[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rrrrrrrrrrrrrrrrrrrr</td>
<td>rrrrrrrrrrrrrrrrrr</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 RXCRC[31:0]: CRC register for receiver

When CRC calculation is enabled, the RXCRC[31:0] bits contain the computed CRC value of the subsequently received bytes. CRC calculation is initialized when the CRCEN bit of SPI_CR1 is set or when a data block is transacted completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on SPI_CRCPOLY register and CRCSIZE bits settings at SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

This bitfield is not used in I2S mode.

RXCRC[31-16] bits are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses.

Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

52.11.13  SPI/I2S underrun data register (SPI_UDRDR)

Address offset: 0x4C
Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled.

<table>
<thead>
<tr>
<th>Bits 31:16 UDRDR[31:16]</th>
<th>Bits 15:0 UDRDR[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

Note: The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.
Bits 31:0 **UDRDR[31:0]**: data at slave underrun condition

The register is taken into account in Slave mode and at underrun condition only. The number of bits considered depends on DSIZE bit settings of the SPI_CFG1 register. Underrun condition handling depends on setting UDRCFG bit at SPI_CFG1 register.

Note: **UDRDR[31-16]** bits are reserved at the peripheral instances with data size limited to 16-bit. There is no constraint when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

### 52.11.14 SPI/I2S configuration register (SPI_I2SCFGR)

Address offset: 0x50

Reset value: 0x0000 0000

This register must be configured when the I2S is disabled (SPE = 0). The content of this register is not taken into account in SPI mode except for the I2SMOD bit which needs to be kept at 0.

This register is reserved at instances not supporting I2S mode.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res</td>
<td>DATFMT</td>
<td>WSNV</td>
<td>FIXCH</td>
<td>CKPOL</td>
<td>CHLEN</td>
<td>DATLEN[1:0]</td>
<td>PCMSYNC</td>
<td>Res</td>
<td>I2SSTD[1:0]</td>
<td>I2SCFG[2:0]</td>
<td>I2SMOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

| Bits 31:26 | Reserved, must be kept at reset value. |
| Bit 25 | **MCKOE**: master clock output enable |
| 0: Master clock output is disabled |
| 1: Master clock output is enabled |
| Bit 24 | **ODD**: odd factor for the prescaler |
| 0: Real divider value is = I2SDIV * 2 |
| 1: Real divider value is = (I2SDIV * 2) + 1 |
| Refer to **Section 52.9.9: Clock generator** for details |
| Bits 23:16 | **I2SDIV[7:0]**: I²S linear prescaler |
| I2SDIV can take any values except the value 1, when ODD is also equal to 1. |
| Refer to **Section 52.9.9: Clock generator** for details |
| Bit 15 | Reserved, must be kept at reset value. |
| Bit 14 | **DATFMT**: data format |
| 0: The data inside the SPI_RXDR or SPI_TXDR are right aligned |
| 1: The data inside the SPI_RXDR or SPI_TXDR are left aligned. |
Bit 13 **WSINV**: word select inversion
This bit is used to invert the default polarity of WS signal.
0: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS falling edge, and the right channel one CK cycle after the WS rising edge.
In MSB or LSB justified mode, the left channel is transferred when WS is HIGH, and the right channel when WS is LOW.
In PCM short mode the data transfer starts at the falling edge of WS, while it starts at the rising edge of WS in PCM long mode.
1: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS rising edge, and the right channel one CK cycle after the WS falling edge.
In MSB or LSB justified mode, the left channel is transferred when WS is LOW, and right channel when WS is HIGH.
In PCM short mode the data transfer starts at the rising edge of WS, while it starts at the falling edge of WS in PCM long mode.

Bit 12 **FIXCH**: fixed channel length in slave
0: the channel length in Slave mode is different from 16 or 32 bits (CHLEN must be set)
1: the channel length in Slave mode is supposed to be 16 or 32 bits (according to CHLEN)

Bit 11 **CKPOL**: serial audio clock polarity
0: the signals generated by the SPI/I2S (i.e. SDO and WS) are changed on the falling edge of CK and the signals received by the SPI/I2S (i.e. SDI and WS) are read of the rising edge of CK.
1: the signals generated by the SPI/I2S (i.e. SDO and WS) are changed on the rising edge of CK and the signals received by the SPI/I2S (i.e. SDI and WS) are read of the falling edge of CK.

Bit 10 **CHLEN**: channel length (number of bits per audio channel)
0: 16-bit wide
1: 32-bit wide

Bits 9:8 **DATLEN[1:0]**: data length to be transferred.
00: 16-bit data length
01: 24-bit data length
10: 32-bit data length
11: Not allowed

*Note: Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), refer to Section 52.3: SPI implementation to check the supported data size.*

Bit 7 **PCMSYNC**: PCM frame synchronization
0: short frame synchronization
1: long frame synchronization

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I2S standard selection
00: I2S Philips standard.
01: MSB justified standard (left justified)
10: LSB justified standard (right justified)
11: PCM standard

For more details on I2S standards, refer to Section 52.9.5: Supported audio protocols.
### Bits 3:1 I2SCFG[2:0]: I2S configuration mode

- 000: slave - transmit
- 001: slave - receive
- 010: master - transmit
- 011: master - receive
- 100: slave - Full Duplex
- 101: master - Full Duplex
- Others, not used

Bit 0 I2SMOD: I2S mode selection
- 0: SPI mode is selected
- 1: I2S/PCM mode is selected

## 52.11.15 SPI/I2S register map

### Table 562. SPI register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>SPI_CR1</td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>SPI_CR2</td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>SPI_CFG1</td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>SPI_CFG2</td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>SPI_IER</td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>SPI_SR</td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>SPI_IFCR</td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>SPI_TXDR</td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x30</td>
<td>SPI_RXDR</td>
<td></td>
</tr>
<tr>
<td>0x34</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x40</td>
<td>SPI_CRCPOLY</td>
<td></td>
</tr>
</tbody>
</table>
Table 562. SPI register map and reset values (continued)

| Offset | Register name | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x44   | SPI_TXCRC     | RXCRC[31-16]¹/² | RXCRC[15-0]³    |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x48   | SPI_RXCRC     | RXCRC[31-16]¹/² | RXCRC[15-0]³    |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x4C   | SPI_UDRDR     | UDRDR[31-16]²/³ | UDRDR[15-0]³    |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x50   | SPI_I2SCFGR   | MCKOE ODD I2SDIV[7:0] | WSTINV FIXCH OXPOL CHLEN PATLEN[1:0] | PCSYNC | TOST[3:0] | I2SCFG2[1:0] | I2SMOD |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

1. The bitfield is reserved for instances with limited set of features and it must be kept at reset value. For more details, refer to the concrete register description in Section 52.11: SPI/I2S registers.

2. The bits 31-16 are reserved for the peripheral instances with data size limited to 16-bit. There is no constrain when the 32-bit access is applied at these addresses. The bits 31-16, when reserved, are always read to zero while any write to them is ignored.

Refer to Section 2.3 for the register boundary addresses.
53  Serial audio interface (SAI)

53.1  Introduction

The SAI interface (serial audio interface) offers a wide set of audio protocols due to its flexibility and wide range of configurations. Many stereo or mono audio applications may be targeted. I2S standards, LSB or MSB-justified, PCM/DSP, TDM, and AC’97 protocols may be addressed for example. SPDIF output is offered when the audio block is configured as a transmitter.

To bring this level of flexibility and reconfigurability, the SAI contains two independent audio subblocks. Each block has its own clock generator and I/O line controller.

The SAI works in master or slave configuration. The audio subblocks are either receiver or transmitter and work synchronously or not (with respect to the other one).

The SAI can be connected with other SAIIs to work synchronously.

53.2  SAI main features

- Two independent audio subblocks which can be transmitters or receivers with their respective FIFO.
- 8-word integrated FIFOs for each audio subblock.
- Synchronous or asynchronous mode between the audio subblocks.
- Possible synchronization between multiple SAIIs.
- Master or slave configuration independent for both audio subblocks.
- Clock generator for each audio block to target independent audio frequency sampling when both audio subblocks are configured in master mode.
- Data size configurable: 8-, 10-, 16-, 20-, 24-, 32-bit.
- Audio protocol: I2S, LSB or MSB-justified, PCM/DSP, TDM, AC’97
- PDM interface, supporting up to 4 microphone pairs
- SPDIF output available if required.
- Up to 16 slots available with configurable size.
- Number of bits by frame can be configurable.
- Frame synchronization active level configurable (offset, bit length, level).
- First active bit position in the slot is configurable.
- LSB first or MSB first for data transfer.
- Mute mode.
- Stereo/Mono audio frame capability.
- Communication clock strobing edge configurable (SCK).
- Error flags with associated interrupts if enabled respectively.
  - Overrun and underrun detection,
  - Anticipated frame synchronization signal detection in slave mode,
  - Late frame synchronization signal detection in slave mode,
  - Codec not ready for the AC’97 mode in reception.
Interrupt sources when enabled:
- Errors,
- FIFO requests.
- 2-channel DMA interface.

53.3 SAI implementation

Table 563. STM32H563/H573 and STM32H562 SAI features (1)

<table>
<thead>
<tr>
<th>SAI features</th>
<th>SAI1</th>
<th>SAI2</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2S, LSB or MSB-justified, PCM/DSP, TDM, AC’97</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FIFO size</td>
<td>8 words</td>
<td>8 words</td>
</tr>
<tr>
<td>SPDIF</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PDM</td>
<td>X(2)</td>
<td>-</td>
</tr>
</tbody>
</table>

1. ‘X’ = supported, ‘-’ = not supported.
2. Only signals D[3:1], and CK[2:1] are available.

53.4 SAI functional description

53.4.1 SAI block diagram

Figure 749 shows the SAI block diagram while Table 564 and Table 565 list SAI internal and external signals.
The SAI is mainly composed of two audio subblocks with their own clock generator. Each audio block integrates a 32-bit shift register controlled by their own functional state machine. Data are stored or read from the dedicated FIFO. FIFO may be accessed by the CPU, or by DMA in order to leave the CPU free during the communication. Each audio block is independent. They can be synchronous with each other.

An I/O line controller manages a set of 4 dedicated pins (SD, SCK, FS, MCLK) for a given audio block in the SAI. Some of these pins can be shared if the two subblocks are declared as synchronous to leave some free to be used as general purpose I/Os. The MCLK pin can be output, or not, depending on the application, the decoder requirement and whether the audio block is configured as the master.

If one SAI is configured to operate synchronously with another one, even more I/Os can be freed (except for pins SD_x).

The functional state machine can be configured to address a wide range of audio protocols. Some registers are present to set-up the desired protocols (audio frame waveform generator).

The audio subblock can be a transmitter or receiver, in master or slave mode. The master mode means the SCK_x bit clock and the frame synchronization signal are generated from the SAI, whereas in slave mode, they come from another external or internal master. There is a particular case for which the FS signal direction is not directly linked to the master or slave mode definition. In AC’97 protocol, it is an SAI output even if the SAI (link controller) is set-up to consume the SCK clock (and so to be in Slave mode).

1. These signals might not be available for all SAI instances. Refer to Section 53.3: SAI implementation for details.
Note: For ease of reading of this section, the notation SAI_x refers to SAI_A or SAI_B, where ‘x’ represents the SAI_A or B subblock.

53.4.2 SAI pins and internal signals

Table 564. SAI internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sai_a_gbl_it/</td>
<td>Output</td>
<td>Audio block A and B global interrupts.</td>
</tr>
<tr>
<td>sai_b_gbl_it</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sai_a_dma,</td>
<td>Input/output</td>
<td>Audio block A and B DMA acknowledges and requests.</td>
</tr>
<tr>
<td>sai_b_dma</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sai_sync_out_sck,</td>
<td>Output</td>
<td>Internal clock and frame synchronization output</td>
</tr>
<tr>
<td>sai_sync_out_fs</td>
<td></td>
<td>signals exchanged with other SAI blocks.</td>
</tr>
<tr>
<td>sai_sync_in_sck,</td>
<td>Input</td>
<td>Internal clock and frame synchronization input</td>
</tr>
<tr>
<td>sai_sync_in_fs</td>
<td></td>
<td>signals exchanged with other SAI blocks.</td>
</tr>
<tr>
<td>sai_a_ker_ck/</td>
<td>Input</td>
<td>Audio block A/B kernel clock.</td>
</tr>
<tr>
<td>sai_b_ker_ck</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sai_pclk</td>
<td>Input</td>
<td>APB clock.</td>
</tr>
</tbody>
</table>

Table 565. SAI input/output pins

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAI_SCK_A/B</td>
<td>Input/output</td>
<td>Audio block A/B bit clock.</td>
</tr>
<tr>
<td>SAI_MCLK_A/B</td>
<td>Output</td>
<td>Audio block A/B master clock.</td>
</tr>
<tr>
<td>SAI_SD_A/B</td>
<td>Input/output</td>
<td>Data line for block A/B.</td>
</tr>
<tr>
<td>SAI_FS_A/B</td>
<td>Input/output</td>
<td>Frame synchronization line for audio block A/B.</td>
</tr>
<tr>
<td>SAI_D[3:1]</td>
<td>Input</td>
<td>PDM bitstream data(1).</td>
</tr>
</tbody>
</table>

1. These signals might not be available in all SAI instances. Refer to Section 53.3: SAI implementation for details.

53.4.3 Main SAI modes

Each audio subblock of the SAI can be configured to be master or slave via MODE bits in the SAI_xCR1 register of the selected audio block.

Master mode

In master mode, the SAI delivers the timing signals to the external connected device:

- The bit clock and the frame synchronization are output on pin SCK_x and FS_x, respectively.
- If needed, the SAI can also generate a master clock on MCLK_x pin.

Both SCK_x, FS_x and MCLK_x are configured as outputs.
Slave mode
The SAI expects to receive timing signals from an external device.
- If the SAI subblock is configured in asynchronous mode, then SCK_x and FS_x pins are configured as inputs.
- If the SAI subblock is configured to operate synchronously with another SAI interface or with the second audio subblock, the corresponding SCK_x and FS_x pins are left free to be used as general purpose I/Os.

In slave mode, MCLK_x pin is not used and can be assigned to another function.
It is recommended to enable the slave device before enabling the master.

Configuring and enabling SAI modes
Each audio subblock can be independently defined as a transmitter or receiver through the MODE bit in the SAI_xCR1 register of the corresponding audio block. As a result, SAI_SD_x pin is respectively configured as an output or an input.

Two master audio blocks in the same SAI can be configured with two different MCLK and SCK clock frequencies. In this case they have to be configured in asynchronous mode.

Each of the audio blocks in the SAI are enabled by SAIEN bit in the SAI_xCR1 register. As soon as this bit is active, the transmitter or the receiver is sensitive to the activity on the clock line, data line and synchronization line in slave mode.

In master TX mode, enabling the audio block immediately generates the bit clock for the external slaves even if there is no data in the FIFO. However FS signal generation is conditioned by the presence of data in the FIFO. After the FIFO receives the first data to transmit, this data is output to external slaves. If there is no data to transmit in the FIFO, 0 values are then sent in the audio frame with an underrun flag generation.

In slave mode, the audio frame starts when the audio block is enabled and when a start of frame is detected.

In Slave TX mode, no underrun event is possible on the first frame after the audio block is enabled, because the mandatory operating sequence in this case is:
1. Write into the SAI_xDR (by software or by DMA).
2. Wait until the FIFO threshold (FLH) flag is different from 0b000 (FIFO empty).
3. Enable the audio block in slave transmitter mode.

53.4.4 SAI synchronization mode
There are two levels of synchronization, either at audio subblock level or at SAI level.

Internal synchronization
An audio subblock can be configured to operate synchronously with the second audio subblock in the same SAI. In this case, the bit clock and the frame synchronization signals are shared to reduce the number of external pins used for the communication. The audio block configured in synchronous mode sees its own SCK_x, FS_x, and MCLK_x pins released back as GPIOs while the audio block configured in asynchronous mode is the one for which FS_x and SCK_x ad MCLK_x I/O pins are relevant (if the audio block is considered as master).
Typically, the audio block in synchronous mode can be used to configure the SAI in full duplex mode. One of the two audio blocks can be configured as a master and the other as slave, or both as slaves with one asynchronous block (corresponding SYNCE[N][1:0] bits set to 00 in SAI_xCR1) and one synchronous block (corresponding SYNCE[N][1:0] bits set to 01 in the SAI_xCR1).

**Note:** Due to internal resynchronization stages, PCLK APB frequency must be higher than twice the bit rate clock frequency.

**External synchronization**

The audio subblocks can also be configured to operate synchronously with another SAI. This can be done as follow:

1. The SAI, which is configured as the source from which the other SAI is synchronized, has to define which of its audio subblock is supposed to provide the FS and SCK signals to other SAI. This is done by programming SYNCOU[1:0] bits.
2. The SAI which receives the synchronization signals, has to select which SAI provides the synchronization by setting the proper value on SYNCEIN[1:0] bits. For each of the two SAI audio subblocks, the user must then specify if it operates synchronously with the other SAI via the SYNCE bit.

**Note:** SYNCEIN[1:0] and SYNCOU[1:0] bits are located into the SAI_GCR register, and SYNCE bits into SAI_xCR1 register.

If both audio subblocks in a given SAI need to be synchronized with another SAI, it is possible to choose one of the following configurations:

- Configure each audio block to be synchronous with another SAI block through the SYNCE[1:0] bits.
- Configure one audio block to be synchronous with another SAI through the SYNCE[1:0] bits. The other audio block is then configured as synchronous with the second SAI audio block through SYNCE[1:0] bits.

The following table shows how to select the proper synchronization signal depending on the SAI block used. For example SAI2 can select the synchronization from SAI1 by setting SAI2 SYNCEIN to 0. If SAI1 wants to select the synchronization coming from SAI2, SAI1 SYNCEIN must be set to 1. Positions noted as ‘Reserved’ must not be used.

<table>
<thead>
<tr>
<th>Block instance</th>
<th>SYNCEIN = 1</th>
<th>SYNCEIN = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAI1</td>
<td>SAI2 sync.</td>
<td>Reserved</td>
</tr>
<tr>
<td>SAI2</td>
<td>Reserved</td>
<td>SAI1 sync.</td>
</tr>
</tbody>
</table>

**53.4.5 Audio data size**

The audio frame can target different data sizes by configuring bit DS[2:0] in the SAI_xCR1 register. The data sizes may be 8, 10, 16, 20, 24 or 32 bits. During the transfer, either the MSB or the LSB of the data are sent first, depending on the configuration of bit LSBFIRST in the SAI_xCR1 register.
53.4.6 Frame synchronization

The FS signal acts as the Frame synchronization signal in the audio frame (start of frame). The shape of this signal is completely configurable in order to target the different audio protocols with their own specificities concerning this Frame synchronization behavior. This reconfigurability is done using register SAI_xFRCR. Figure 750 illustrates this flexibility.

Figure 750. Audio frame

In AC’97 mode or in SPDIF mode (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01 in the SAI_xCR1 register), the frame synchronization shape is forced to match the AC’97 protocol. The SAI_xFRCR register value is ignored.

Each audio block is independent and consequently each one requires a specific configuration.

Frame length

- Master mode
  The audio frame length can be configured to up to 256 bit clock cycles, by setting FRL[7:0] field in the SAI_xFRCR register.

  If the frame length is greater than the number of declared slots for the frame, the remaining bits to transmit is extended to 0 or the SD line is released to HI-z depending the state of bit TRIS in the SAI_xCR2 register (refer to FS signal role). In reception mode, the remaining bit is ignored.

  If bit NODIV is cleared, (FRL+1) must be equal to a power of 2, from 8 to 256, to ensure that an audio frame contains an integer number of MCLK pulses per bit clock cycle.

  If bit NODIV is set, the (FRL+1) field can take any value from 8 to 256. Refer to Section 53.4.8: SAI clock generator.

- Slave mode
  The audio frame length is mainly used to specify to the slave the number of bit clock cycles per audio frame sent by the external master. It is used mainly to detect from the master any anticipated or late occurrence of the Frame synchronization signal during an ongoing audio frame. In this case an error is generated. For more details refer to Section 53.4.14: Error flags.

  In slave mode, there are no constraints on the FRL[7:0] configuration in the SAI_xFRCR register.

  The number of bits in the frame is equal to FRL[7:0] + 1.

  The minimum number of bits to transfer in an audio frame is 8.
Frame synchronization polarity

FSPOL bit in the SAI_xFRCR register sets the active polarity of the FS pin from which a frame is started. The start of frame is edge sensitive.

In slave mode, the audio block waits for a valid frame to start transmitting or receiving. Start of frame is synchronized to this signal. It is effective only if the start of frame is not detected during an ongoing communication and assimilated to an anticipated start of frame (refer to Section 53.4.14: Error flags).

In master mode, the frame synchronization is sent continuously each time an audio frame is complete until the SAIEN bit in the SAI_xCR1 register is cleared. If no data are present in the FIFO at the end of the previous audio frame, an underrun condition is managed as described in Section 53.4.14: Error flags), but the audio communication flow is not interrupted.

Frame synchronization active level length

The FSALL[6:0] bits of the SAI_xFRCR register enable the configuration of the length of the active level of the Frame synchronization signal. The length can be set from 1 to 128 bit clock cycles.

As an example, the active length can be half of the frame length in I2S, LSB or MSB-justified modes, or one-bit wide for PCM/DSP or TDM.

Frame synchronization offset

Depending on the audio protocol targeted in the application, the Frame synchronization signal can be asserted when transmitting the last bit or the first bit of the audio frame (this is the case in I2S standard protocol and in MSB-justified protocol, respectively). FSOFF bit in the SAI_xFRCR register enables to choose one of the two configurations.

FS signal role

The FS signal can have a different meaning depending on the FS function. FSDEF bit in the SAI_xFRCR register selects which meaning it has:

- 0: start of frame, like for instance the PCM/DSP, TDM, AC’97, audio protocols,
- 1: start of frame and channel side identification within the audio frame like for the I2S, the MSB or LSB-justified protocols.

When the FS signal is considered as a start of frame and channel side identification within the frame, the number of declared slots must be considered to be half the number for the left channel and half the number for the right channel. If the number of bit clock cycles on half audio frame is greater than the number of slots dedicated to a channel side, and TRIS = 0, 0 is sent for transmission for the remaining bit clock cycles in the SAI_xCR2 register. Otherwise if TRIS = 1, the SD line is released to HI-Z. In reception mode, the remaining bit clock cycles are not considered until the channel side changes.
1. The frame length must be even.

   If FSDEF bit in SAI_xFRCR is kept clear, so FS signal is equivalent to a start of frame, and if the number of slots defined in NBSLOT[3:0] in SAI_xSLOTR multiplied by the number of bits by slot configured in SLOTSZ[1:0] in SAI_xSLOTR is less than the frame size (bit FRL[7:0] in the SAI_xFRCR register), then:
   
   - if TRIS = 0 in the SAI_xCR2 register, the remaining bit after the last slot is forced to 0 until the end of frame in case of transmitter,
   - if TRIS = 1, the line is released to HI-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.
Figure 752. FS role is start of frame (FSDEF = 0)

The FS signal is not used when the audio block in transmitter mode is configured to get the SPDIF output on the SD line. The corresponding FS I/O is released and left free for other purposes.

53.4.7 Slot configuration

The slot is the basic element in the audio frame. The number of slots in the audio frame is equal to NBSLOT[3:0] + 1.

The maximum number of slots per audio frame is fixed at 16.

For AC’97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

Each slot can be defined as a valid slot, or not, by setting SLOTEN[15:0] bits of the SAI_xSLOTR register.

When a invalid slot is transferred, the SD data line is either forced to 0 or released to HI-z depending on TRIS bit configuration (refer to Output data line management on an inactive slot) in transmitter mode. In receiver mode, the received value from the end of this slot is ignored. Consequently, there is no FIFO access and so no request to read or write the FIFO linked to this inactive slot status.

The slot size is also configurable as shown in Figure 753. The size of the slots is selected by setting SLOTSZ[1:0] bits in the SAI_xSLOTR register. The size is applied identically for each slot in an audio frame.
It is possible to choose the position of the first data bit to transfer within the slots. This offset is configured by FBOFF[4:0] bits in the SAI_xSLOTR register. 0 values are injected in transmitter mode from the beginning of the slot until this offset position is reached. In reception, the bit in the offset phase is ignored. This feature targets the LSB justified protocol (if the offset is equal to the slot size minus the data size).

It is mandatory to respect the following conditions to avoid bad SAI behavior:

\[ FBOFF \leq (\text{SLOTSZ} - \text{DS}), \]
\[ DS \leq \text{SLOTSZ}, \]
\[ \text{NBSLOT} \times \text{SLOTSZ} \leq \text{FRL} \text{ (frame length)}, \]

The number of slots must be even when bit FSDEF in the SAI_xFRCR register is set.

In AC’97 and SPDIF protocol (bit PRTC[1:0] = 10 or PRTC[1:0] = 01), the slot size is automatically set as defined in Section 53.4.11: AC’97 link controller.
53.4.8 SAI clock generator

Each audio block has its own clock generator. The clock generator builds the master clock (MCLK_x) and bit clock (SCK_x) signals from the sai_x_ker_ck. The sai_x_ker_ck clock is delivered by the clock controller of the product (RCC).

Generation of the master clock (MCLK_x)

The clock generator provides the master clock (MCLK_x) when the audio block is defined as Master or Slave. The master clock is generated as soon as the MCKEN bit is set to 1 even if the SAIEN bit for the corresponding block is set to 0. This feature can be useful if the MCLK_x clock is used as system clock for an external audio device, since it enables the generation of the MCLK_x before activating the audio stream.

To generate a master clock on MCLK_x output before transferring the audio samples, the user application has to follow the sequence below:

1. Check that SAIEN = 0.
2. Program the MCKDIV[5:0] divider to the required value.
3. Set the MCKEN bit to 1.
4. Later, the application can configure other parts of the SAI, and sets the SAIEN bit to 1 to start the transfer of audio samples.

To avoid disturbances on the clock generated on MCLK_x output, the following operations are not recommended:

- Changing MCKDIV when MCKEN = 1
- Setting MCKEN to 0 if the SAIEN = 1

The SAI guarantees that there is no spurs on MCLK_x output when the MCLK_x is switched ON and OFF via MCKEN bit (with SAIEN = 0).

Table 567 shows MCLK_x activation conditions.

<table>
<thead>
<tr>
<th>MCLKEN</th>
<th>NODIV</th>
<th>SAIEN for block x</th>
<th>MCLK_x</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>Enabled</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Enabled</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>Enabled</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td></td>
<td>Enabled</td>
</tr>
</tbody>
</table>

Note: MCLK_x can also be generated in AC’97 mode, when MCLKEN is set to 1.
Generation of the bit clock (SCK_x)

The clock generator provides the bit clock (SCK_x) when the audio block is defined as Master. The frame synchronization (FS_x) is also derived from the signals provided by the clock generator.

In Slave mode, the value of NODIV and OSR fields are ignored, and the SCK_x clock is not generated.

The bit clock strobing edge of SCK_x can be configured through the CKSTR fields, which is functional both in master and slave mode.

Figure 755 illustrates the architecture of the audio block clock generator.

![Figure 755. Audio block clock generator overview](image)

The NODIV bit must be used to force the ratio between the master clock (MCLK_x) and the frame synchronization (FS_x) frequency to 256 or 512.

- If NODIV is set to 0, the frequency ratio between the frame synchronization and the master clock is fixed to 512 or 256, according to OSR value, but the frame length must be a power of 2. More details are given hereafter.
- If NODIV is set to 1, the application can adjust the frequency of the bit clock (SCK_x) via MCKDIV. In addition there is no restriction on the frame length value as long as the frame length is bigger or equal to 8 (i.e. FRL[7:0] > 6). The frame synchronization frequency depends on MCKDIV and frame length (FRL[7:0]). In that case, the frequency of the MCLK_x is equal to the SCK_x.

The NODIV, MCKEN, SAIEN, OVR, CKSTR and MCKDIV[5:0] bits belong to the SAI_xCR1 register, while FRL[7:0] belongs to SAI_xFRCR.
Clock generator programming when NODIV = 0

In that case, MCLK_x frequency is:
- \( F_{MCLK_x} = 256 \times F_{FS_x} \) if OSR = 0
- \( F_{MCLK_x} = 512 \times F_{FS_x} \) if OSR = 1

When MCKDIV is different from 0, MCLK_x frequency is given by the formula below:

\[
F_{MCLK_x} = \frac{F_{sai_x \_ker \_ck}}{MCKDIV}
\]

The frame synchronization frequency is given by:

\[
F_{FS_x} = \frac{F_{sai_x \_ker \_ck}}{MCKDIV \times (OSR + 1) \times 256}
\]

The bit clock frequency (SCK_x) is given by the following formula:

\[
F_{SCK_x} = \frac{F_{sai_x \_ker \_ck} \times (FRL + 1)}{MCKDIV \times (OSR + 1) \times 256}
\]

**Note:** When NODIV is equal to 0, \((FRL + 1)\) must be a power of two. In addition \((FRL + 1)\) must range between 8 and 256. \((FRL + 1)\) represents the number of bit clock in the audio frame.

When MCKDIV division ratio is odd, the MCLK duty cycle is not 50%. The bit clock signal (SCK_x) can also have a duty cycle different from 50% if MCKDIV is odd, if OSR is equal to 0, and if \((FRL + 1) = 2^8\).

It is recommended, to program MCKDIV to an even value or to big values (higher than 10).

**Note that MCKDIV = 0 gives the same result as MCKDIV = 1.**

Clock generator programming when NODIV = 1

When MCKDIV is different from 0, the frequency of the bit clock (SCK_x) is given in the formula below:

\[
F_{SCK_x} = F_{MCLK_x} = \frac{F_{sai_x \_ker \_ck}}{MCKDIV}
\]

The frequency of the frame synchronization (FS_x) is given by the following formula:

\[
F_{FS_x} = \frac{F_{sai_x \_ker \_ck}}{(FRL + 1) \times MCKDIV}
\]

**Note:** When NODIV is set to 1, \((FRL + 1)\) can take any values from 8 to 256.

**Note that MCKDIV = 0 gives the same result as MCKDIV = 1.**
Clock generator programming examples

Table 568 gives programming examples for 48, 96 and 192 kHz.

<table>
<thead>
<tr>
<th>Input clock frequency</th>
<th>MCLK</th>
<th>F(<em>{MCLK}/F(</em>{FS})</th>
<th>FRL</th>
<th>OSR</th>
<th>NODIV</th>
<th>MCKEN</th>
<th>MCKDIV[5:0]</th>
<th>Audio Sampling frequency (F(_{FS}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>sai_x_ker_ck</td>
<td>98.304 MHz</td>
<td>512</td>
<td>(2^{N-1})</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0 or 1</td>
<td>192 kHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td>(2^{N-1})</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>96 kHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512</td>
<td>(2^{N-1})</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>48 kHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256</td>
<td>(2^{N-1})</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>192 kHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256</td>
<td>(2^{N-1})</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>96 kHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256</td>
<td>(2^{N-1})</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>48 kHz</td>
</tr>
<tr>
<td></td>
<td>Y</td>
<td>-</td>
<td>63</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>192 kHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
<td>63</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>16</td>
<td>96 kHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
<td>63</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>32</td>
<td>48 kHz</td>
</tr>
</tbody>
</table>

1. \(N\) is an integer value between 3 and 8.

53.4.9 Internal FIFOs

Each audio block in the SAI has its own FIFO. Depending if the block is defined to be a transmitter or a receiver, the FIFO can be written or read, respectively. There is therefore only one FIFO request linked to FREQ bit in the SAI_xSR register.

An interrupt is generated if FREQIE bit is enabled in the SAI_xIM register. This depends on:
- FIFO threshold setting (FLVL bits in SAI_xCR2)
- Communication direction (transmitter or receiver). Refer to Interrupt generation in transmitter mode and Interrupt generation in reception mode.

**Interrupt generation in transmitter mode**

The interrupt generation depends on the FIFO configuration in transmitter mode:
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit set by hardware to 1 in SAI_xSR register) if no data are available in SAI_xDR register (FLVL[2:0] bits in SAI_xSR is less than 001b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO is no more empty (FLVL[2:0] bits in SAI_xSR are different from 0b000) i.e one or more data are stored in the FIFO.
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 001b), an interrupt is generated (FREQ bit set by hardware to 1 in SAI_xSR register) if less than a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b010). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when at least a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher or equal to 0b010).
- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit set by hardware to 1 in...
SAI_xSR register) if less than half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 011b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when at least half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher or equal to 011b).

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO three quarter full (FTH[2:0] set to 011b), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if less than three quarters of the FIFO contain data (FLVL[2:0] bits in SAI_xSR are less than 011b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when at least three quarters of the FIFO contain data (FLVL[2:0] bits in SAI_xSR are higher or equal to 011b).

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if the FIFO is not full (FLVL[2:0] bits in SAI_xSR is less than 0b101). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI_xSR is equal to 101b value).

**Interrupt generation in reception mode**

The interrupt generation depends on the FIFO configuration in reception mode:

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b00), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least one data is available in SAI_xDR register (FLVL[2:0] bits in SAI_xSR is higher or equal to 001b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO becomes empty (FLVL[2:0] bits in SAI_xSR is equal to 0b00) i.e no data are stored in FIFO.

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO quarter fully (FTH[2:0] set to 001b), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least one quarter of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR is higher or equal to 0b010). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when less than a quarter of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR is less than 0b001).

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO half fully (FTH[2:0] set to 0b010 value), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least half of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR is higher or equal to 011b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when less than half of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR is less than 0b110).

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO three quarter full (FTH[2:0] set to 011b value), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if at least three quarters of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR is higher or equal to 0b100). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO has less than three quarters of the FIFO data locations available (FLVL[2:0] bits in SAI_xSR is less than 0b010).

- When the FIFO threshold bits in SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in SAI_xSR register) if the FIFO is full (FLVL[2:0] bits in SAI_xSR is equal to 101b). This Interrupt (FREQ bit in SAI_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI_xSR is less than 101b).
Like interrupt generation, the SAI can use the DMA if DMAEN bit in the SAI_xCR1 register is set. The FREQ bit assertion mechanism is the same as the interrupt generation mechanism described above for FREQIE.

Each FIFO is an 8-word FIFO. Each read or write operation from/to the FIFO targets one word FIFO location whatever the access size. Each FIFO word contains one audio slot. FIFO pointers are incremented by one word after each access to the SAI_xDR register.

Data must be right aligned when it is written in the SAI_xDR.

Data received are right aligned in the SAI_xDR.

The FIFO pointers can be reinitialized when the SAI is disabled by setting bit FFLUSH in the SAI_xCR2 register. If FFLUSH is set when the SAI is enabled the data present in the FIFO are lost automatically.

### 53.4.10 PDM interface

The PDM (Pulse Density Modulation) interface is provided in order to support digital microphones. Up to 4 digital microphone pairs can be connected in parallel. Depending on product implementation, less microphones can be supported (refer to Section 53.3: SAI implementation).

*Figure 756 shows a typical connection of a digital microphone pair via a PDM interface. Both microphones share the same bitstream clock and data line. Thanks to a configuration pin (LR), a microphone can provide valid data on SAI_CK[m] rising edge while the other provides valid data on SAI_CK[m] falling edge (m being the number of clock lines).*

*Figure 756. PDM typical connection and timing*

1. \( n \) refers to the number of data lines and \( p \) to the number of microphone pairs.

The PDM function is intended to be used in conjunction with SAI_A subblock configured in TDM master mode. It cannot be used with SAI_B subblock. The PDM interface uses the timing signals provided by the TDM interface of SAI_A and adapts them to generate a bitstream clock (SAI_CK[m]).
The data processing sequence into the PDM is the following:

1. The PDM interface builds the bitstream clock from the bit clock received from the TDM interface of SAI_A.
2. The bitstream data received from the microphones (SAI_D[n]) are de-interleaved and go through a 7-bit delay line in order to fine-tune the delay of each microphone with the accuracy of the bitstream clock.
3. The shift registers translate each serial bitstream into bytes.
4. The last operation consists in shifting-out the resulting bytes to SAI_A via the serial data line of the TDM interface.

*Figure 757* hereafter shows the block diagram of PDM interface, with a detailed view of a de-interleaver.

*Note:* The PDM interface does not embed the decimation filter required to build-up the PCM audio samples from the bitstream. It is up to the application software to perform this operation.

*Figure 757. Detailed PDM interface block diagram*

1. \( n \) refers to the number of data lines and \( p \) to the number of microphone pairs.
2. These signals might not be available in all SAI instances. Refer to *Section 53.3: SAI implementation* for details.

The PDM interface can be enabled through the PDMEN bit in SAI_PDMCR register. However the PDM interface must be enabled prior to enabling SAI_A block.
To reduce the memory footprint, the user can select the amount of microphones the application needs. This can be done through MICNBR[1:0] bits. It is possible to select between 2, 4, 6 or 8 microphones. For example, if the application is using 3 microphones, the user has to select 4.

**Enabling the PDM interface**

To enable the PDM interface, follow the sequence below:

1. Configure SAI_A in TDM master mode (see Table 569).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via PDMEN bit.
4. Enable the SAI_A.

**Note:** Once the PDM interface and SAI_A are enabled, the first 2 TDMA frames received on SAI_ADR are invalid and must be dropped.

**Start-up sequence**

*Figure 758* shows the start-up sequence: Once the PDM interface is enabled, it waits for the frame synchronization event prior to starting the acquisition of the microphone samples. After 8 SAI_CK clock periods, a data byte coming from each microphone is available, and transferred to the SAI, via the TDM interface.

*Figure 758. Start-up sequence*
SAI_ADR data format

The arrangement of the data coming from the microphone into the SAI_ADR register depends on the following parameters:

- The amount of microphones
- The slot width selected
- LSBFIRST bit.

The slot width defines the amount of significant bits into each word available into the SAI_ADR.

When a slot width of 32 bits is selected, each data available into the SAI_ADR contains 32 useful bits. This reduces the amount of words stored into the memory. However the counterpart is that the software has to perform some operations to de-interleave the data of each microphone.

In the other hand, when the slot width is set to 8 bits, each data available into the SAI_ADR contain 8 useful bits. This increases the amount of words stored into the memory. However, it offers the advantage to avoid extra processing since each word contains information from one microphone.

SAI_ADR data format example

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to Figure 759.

  For an 8 microphone configuration, two consecutive words read from the SAI_ADR register contain a data byte from each microphone.

  For a 4 microphones configuration, each word read from the SAI_ADR register contains a data byte from each microphone.

**Figure 759. SAI_ADR format in TDM, 32-bit slot width**

- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to Figure 760.

  For an 8 microphone configuration, four consecutive words read from the SAI_ADR register contain a data byte from each microphone. Note that the 16-bit data of SAI_ADR are right aligned.

  For 4 or 2 microphone configuration, the SAI behavior is similar to 8-microphone configurations. Up to 2 words of 16 bits are required to acquire a byte from 4 microphones and a single word for 2 microphones.
Using a 8-bit slot width (DS = 0b010 and SLOTSZ = 0). Refer to Figure 761. For an 8 microphone configuration, 8 consecutive words read from the SAI_ADR register contain a byte of data from each microphone. Note that the 8-bit data of SAI_ADR are right aligned.

For 4 or 2 microphone configuration, the SAI behavior is similar to 8 microphone configurations. Up to 4 words of 8 bits are required to acquire a byte from 4 microphones and 2 words from 2 microphones.
TDM configuration for PDM interface

SAI_A TDM interface is internally connected to the PDM interface to get the microphone samples. The user application must configure the PDM interface as shown in Table 569 to ensure a good connection with the PDM interface.

Table 569. TDM settings

<table>
<thead>
<tr>
<th>Bit Fields</th>
<th>Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE</td>
<td>0b01</td>
<td>Mode must be MASTER receiver</td>
</tr>
<tr>
<td>PRTCFG</td>
<td>0b00</td>
<td>Free protocol for TDM</td>
</tr>
<tr>
<td>DS</td>
<td>X</td>
<td>To be adjusted according to the required data format, in accordance to the frame length and the number of slots (FRL and NBSLOT). See Table 570.</td>
</tr>
<tr>
<td>LSBFIRST</td>
<td>X</td>
<td>This parameter can be used according to the wanted data format</td>
</tr>
<tr>
<td>CKSTR</td>
<td>0</td>
<td>Signal transitions occur on the rising edge of the SCK_A bit clock. Signals are stable on the falling edge of the bit clock.</td>
</tr>
<tr>
<td>MONO</td>
<td>0</td>
<td>Stereo mode</td>
</tr>
<tr>
<td>FRL</td>
<td>X</td>
<td>To be adjusted according to the number of microphones (MICNBR). See Table 570.</td>
</tr>
<tr>
<td>FSALL</td>
<td>0</td>
<td>Pulse width is one bit clock cycle</td>
</tr>
<tr>
<td>FSDEF</td>
<td>0</td>
<td>FS signal is a start of frame</td>
</tr>
</tbody>
</table>
Adjusting the bitstream clock rate

To properly program the SAI TDM interface, the user application must take into account the settings given in Table 569, and follow the below sequence:

1. Adjust the bit clock frequency \( F_{SCK_A} \) according to the required frequency for the PDM bitstream clock, using the following formula:

\[
F_{SCK_A} = F_{PDM\_CK} \times (MICNBR + 1) \times 2
\]

MICNBR can be 0, 1, 2 or 3 (0 = 2 microphones., see Section 53.6.18)

2. Set the frame length (FRL) using the following formula

\[
FRL = (16 \times (MICNBR + 1)) - 1
\]

3. Configure the slot size (DS) to a multiple of (FRL+1).

### Table 569. TDM settings (continued)

<table>
<thead>
<tr>
<th>Bit Fields</th>
<th>Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSPOL</td>
<td>1</td>
<td>FS is active High</td>
</tr>
<tr>
<td>FSOFF</td>
<td>0</td>
<td>FS is asserted on the first bit of slot 0</td>
</tr>
<tr>
<td>FBOFF</td>
<td>0</td>
<td>No offset on slot</td>
</tr>
<tr>
<td>SLOTSZ</td>
<td>0</td>
<td>Slot size = data size</td>
</tr>
<tr>
<td>NBSLOT</td>
<td>X</td>
<td>To be adjusted according to the required data format, in accordance to the slot size, and the frame length (FRL and DS). See Table 570.</td>
</tr>
<tr>
<td>SLOTEN</td>
<td>X</td>
<td>To be adjusted according to NBSLOT</td>
</tr>
<tr>
<td>NODIV</td>
<td>1</td>
<td>No need to generate a master clock MCLK</td>
</tr>
<tr>
<td>MCKDIV</td>
<td>X</td>
<td>Depends on the frequency provided to sai_a_ker_ck input. This parameter must be adjusted to generate the proper bitstream clock frequency. See Table 570.</td>
</tr>
</tbody>
</table>
Adjusting the delay lines

When the PDM interface is enabled, the application can adjust on-the-fly the delay cells of each microphone input via SAI_PDMDLY register.

The new delays values become effective after two TDM frames.
53.4.11 AC’97 link controller

The SAI is able to work as an AC’97 link controller. In this protocol:

- The slot number and the slot size are fixed.
- The frame synchronization signal is perfectly defined and has a fixed shape.

To select this protocol, set PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- NBSLOT[3:0] and SLOTSZ[1:0] bits are consequently ignored.
- The number of slots is fixed to 13 slots. The first one is 16-bit wide and all the others are 20-bit wide (data slots).
- FBOFF[4:0] bits in the SAI_xSLOTR register are ignored.
- The SAI_xFRCR register is ignored.
- The MCLK is not used.

The FS signal from the block defined as asynchronous is configured automatically as an output, since the AC’97 controller link drives the FS signal whatever the master or slave configuration.

*Figure 762 shows an AC’97 audio frame structure.*

**Figure 762. AC’97 audio frame**

**Note:** In AC’97 protocol, bit 2 of the tag is reserved (always 0), so bit 2 of the TAG is forced to 0 level whatever the value written in the SAI FIFO.

For more details about tag representation, refer to the AC’97 protocol standard.

One SAI can be used to target an AC’97 point-to-point communication.

Using two SAIs (for devices featuring two embedded SAIs) enables the control of three external AC’97 decoders as illustrated in *Figure 763.*

In SAI1, the audio block A must be declared as asynchronous master transmitter whereas the audio block B is defined to be slave receiver and internally synchronous to the audio block A.

The SAI2 is configured for audio block A and B both synchronous with the external SAI1 in slave receiver mode.
In receiver mode, the SAI acting as an AC’97 link controller requires no FIFO request and so no data storage in the FIFO when the Codec ready bit in the slot 0 is decoded low. If bit CNRDYIE is enabled in the SAI_xIM register, flag CNRDY is set in the SAI_xSR register and an interrupt is generated. This flag is dedicated to the AC’97 protocol.

**Clock generator programming in AC’97 mode**

In AC’97 mode, the frame length is fixed at 256 bits, and its frequency must be set to 48 kHz. The formulas given in Section 53.4.8: SAI clock generator must be used with FRL = 255, in order to generate the proper frame rate (FFS_x).

---

Figure 763. Example of typical AC’97 configuration on devices featuring at least 2 embedded SAIs (three external AC’97 decoders)
53.4.12  SPDIF output

The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958. To select SPDIF mode, set PRTCFG[1:0] bit to 01 in the SAI_xCR1 register.

For SPDIF protocol:

- Only SD data line is enabled.
- FS, SCK, MCLK I/Os pins are left free.
- MODE[1] bit is forced to 0 to select the master mode in order to enable the clock generator of the SAI and manage the data rate on the SD line.
- The data size is forced to 24 bits. The value set in DS[2:0] bits in the SAI_xCR1 register is ignored.
- The clock generator must be configured to define the symbol-rate, knowing that the bit clock must be twice the symbol-rate. The data is coded in Manchester protocol.
- The SAI_xFRCR and SAI_xSLOTR registers are ignored. The SAI is configured internally to match the SPDIF protocol requirements as shown in Figure 764.

**Figure 764. SPDIF format**

A SPDIF block contains 192 frames. Each frame is composed of two 32-bit sub-frames, generally one for the left channel and one for the right channel. Each sub-frame is composed of a SOPD pattern (4-bit) to specify if the sub-frame is the start of a block (and so is identifying a channel A) or if it is identifying a channel A somewhere in the block, or if it is referring to channel B (see Table 571). The next 28 bits of channel information are composed of 24 bits data + 4 status bits.
The data stored in SAI_xDR has to be filled as follows:
- SAI_xDR[26:24] contain the Channel status, User and Validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, then data must be mapped on SAI_xDR[23:4].
If the data size is 16 bits, then data must be mapped on SAI_xDR[23:8].
SAI_xDR[23] always represents the MSB.

**Table 571. SOPD pattern**

<table>
<thead>
<tr>
<th>SOPD</th>
<th>Preamble coding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>last bit is 0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>11101000</td>
<td>00010111 Channel A data at the start of block</td>
</tr>
<tr>
<td>W</td>
<td>11100100</td>
<td>00011011 Channel B data somewhere in the block</td>
</tr>
<tr>
<td>M</td>
<td>11100010</td>
<td>00011101 Channel A data</td>
</tr>
</tbody>
</table>

The data stored in SAI_xSR has to be filled as follows:

- SAI_xDR[26:24] contain the Channel status, User and Validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, then data must be mapped on SAI_xDR[23:4].
If the data size is 16 bits, then data must be mapped on SAI_xDR[23:8].
SAI_xDR[23] always represents the MSB.

**Figure 765. SAI_xDR register ordering**

Note: The transfer is performed always with LSB first.

The SAI first sends the adequate preamble for each sub-frame in a block. The SAI_xDR is then sent on the SD line (manchester coded). The SAI ends the sub-frame by transferring the Parity bit calculated as described in **Table 572**.

**Table 572. Parity bit calculation**

<table>
<thead>
<tr>
<th>SAI_xDR[26:0]</th>
<th>Parity bit P value transferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>odd number of 0</td>
<td>0</td>
</tr>
<tr>
<td>odd number of 1</td>
<td>1</td>
</tr>
</tbody>
</table>

The underrun is the only error flag available in the SAI_xSR register for SPDIF mode since the SAI can only operate in transmitter mode. As a result, the following sequence must be...
executed to recover from an underrun error detected via the underrun interrupt or the underrun status bit:

1. Disable the DMA stream (via the DMA peripheral) if the DMA is used.
2. Disable the SAI and check that the peripheral is physically disabled by polling the SAIEN bit in SAI_xCR1 register.
3. Clear the COVRUNDR flag in the SAI_xCLRFR register.
4. Flush the FIFO by setting the FFLUSH bit in SAI_xCR2.
   The software needs to point to the address of the future data corresponding to a start of new block (data for preamble B). If the DMA is used, the DMA source base address pointer must be updated accordingly.
5. Enable again the DMA stream (DMA peripheral) if the DMA used to manage data transfers according to the new source base address.
6. Enable again the SAI by setting SAIEN bit in SAI_xCR1 register.

Clock generator programming in SPDIF generator mode

For the SPDIF generator, the SAI provides a bit clock twice faster as the symbol-rate. The table hereafter shows usual examples of symbol rates with respect to the audio sampling rate.

<table>
<thead>
<tr>
<th>Audio sampling frequencies ($F_S$)</th>
<th>Symbol-rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>44.1 kHz</td>
<td>2.8224 MHz</td>
</tr>
<tr>
<td>48 kHz</td>
<td>3.072 MHz</td>
</tr>
<tr>
<td>96 kHz</td>
<td>6.144 MHz</td>
</tr>
<tr>
<td>192 kHz</td>
<td>12.288 MHz</td>
</tr>
</tbody>
</table>

More generally, the relationship between the audio sampling frequency ($F_S$) and the bit clock rate ($F_{SCK_x}$) is given by the formula:

$$F_S = \frac{F_{SCK_x}}{128}$$

The bit clock rate is obtained as follows:

$$F_{SCK_x} = \frac{F_{sai_x \_ker \_ck}}{MCKDIV}$$

**Note:** The above formulas are valid only if NODIV is set to 1 in SAI_ACR1 register.
53.4.13 Specific features

The SAI interface embeds specific features which can be useful depending on the audio protocol selected. These functions are accessible through specific bits of the SAI_xCR2 register.

**Mute mode**

The mute mode can be used when the audio subblock is a transmitter or a receiver.

**Audio subblock in transmission mode**

In transmitter mode, the mute mode can be selected at anytime. The mute mode is active for entire audio frames. The MUTE bit in the SAI_xCR2 register enables the mute mode when it is set during an ongoing frame.

The mute mode bit is strobed only at the end of the frame. If it is set at this time, the mute mode is active at the beginning of the new audio frame and for a complete frame, until the next end of frame. The bit is then strobed to determine if the next frame is still a mute frame.

If the number of slots set through NBSLOT[3:0] bits in the SAI_xSLOTR register is lower than or equal to 2, it is possible to specify if the value sent in mute mode is 0 or if it is the last value of each slot. The selection is done via MUTEVAL bit in the SAI_xCR2 register.

If the number of slots set in NBSLOT[3:0] bits in the SAI_xSLOTR register is greater than 2, MUTEVAL bit in the SAI_xCR2 is meaningless as 0 values are sent on each bit on each slot.

The FIFO pointers are still incremented in mute mode. This means that data present in the FIFO and for which the mute mode is requested are discarded.

**Audio subblock in reception mode**

In reception mode, it is possible to detect a mute mode sent from the external transmitter when all the declared and valid slots of the audio frame receive 0 for a given consecutive number of audio frames (MUTECNT[5:0] bits in the SAI_xCR2 register).

When the number of MUTE frames is detected, the MUTEDET flag in the SAI_xSR register is set and an interrupt can be generated if MUTEDETIE bit is set in SAI_xCR2.

The mute frame counter is cleared when the audio subblock is disabled or when a valid slot receives at least one data in an audio frame. The interrupt is generated just once, when the counter reaches the value specified in MUTECNT[5:0] bits. The interrupt event is then reinitialized when the counter is cleared.

*Note:* The mute mode is not available for SPDIF audio blocks.

**Mono/stereo mode**

In transmitter mode, the mono mode can be addressed, without any data preprocessing in memory, assuming the number of slots is equal to 2 (NBSLOT[3:0] = 0001 in SAI_xSLOTR).

In this case, the access time to and from the FIFO is reduced by 2 since the data for slot 0 is duplicated into data slot 1.

To enable the mono mode,

1. Set MONO bit to 1 in the SAI_xCR1 register.
2. Set NBSLOT to 1 and SLOTEN to 3 in SAI_xSLOTR.
In reception mode, the MONO bit can be set and is meaningful only if the number of slots is equal to 2 as in transmitter mode. When it is set, only slot 0 data are stored in the FIFO. The data belonging to slot 1 are discarded since, in this case, it is supposed to be the same as the previous slot. If the data flow in reception mode is a real stereo audio flow with a distinct and different left and right data, the MONO bit is meaningless. The conversion from the output stereo file to the equivalent mono file is done by software.

**Companding mode**

Telecommunication applications can require to process the data to be transmitted or received using a data companding algorithm.

Depending on the COMP[1:0] bits in the SAI_xCR2 register (used only when Free protocol mode is selected), the application software can choose to process or not the data before sending it on SD serial output line (compression) or to expand the data after the reception on SD serial input line (expansion) as illustrated in Figure 766. The two companding modes supported are the µ-Law and the A-Law log which are a part of the CCITT G.711 recommendation.

The companding standard used in the United States and Japan is the µ-Law. It supports 14 bits of dynamic range (COMP[1:0] = 10 in the SAI_xCR2 register).

The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

Both µ-Law or A-Law companding standard can be computed based on 1’s complement or 2’s complement representation depending on the CPL bit setting in the SAI_xCR2 register.

In µ-Law and A-Law standards, data are coded as 8 bits with MSB alignment. Companded data are always 8-bit wide. For this reason, DS[2:0] bits in the SAI_xCR1 register are forced to 010 when the SAI audio block is enabled (SAIEN bit = 1 in the SAI_xCR1 register) and when one of these two companding modes selected through the COMP[1:0] bits.

If no companding processing is required, COMP[1:0] bits must be kept clear.
Expansion and compression mode are automatically selected through the SAI_xCR2:

- If the SAI audio block is configured to be a transmitter, and if the COMP[1] bit is set in the SAI_xCR2 register, the compression mode is applied.
- If the SAI audio block is declared as a receiver, the expansion algorithm is applied.

**Output data line management on an inactive slot**

In transmitter mode, it is possible to choose the behavior of the SD line output when an inactive slot is sent on the data line (via TRIS bit).

- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in HI-z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which may result in a short circuit. To ensure a gap between transmissions, if the data is lower than 32-bit, the data can be extended to 32-bit by setting bit SLOTSZ[1:0] = 10 in the SAI_xSLOTR register. The SD output pin is then tri-stated at the end of the LSB of the active slot (during the padding to 0 phase to extend the data to 32-bit) if the following slot is declared inactive.

In addition, if the number of slots multiplied by the slot size is lower than the frame length, the SD output line is tri-stated when the padding to 0 is done to complete the audio frame. *Figure 767* illustrates these behaviors.
When the selected audio protocol uses the FS signal as a start of frame and a channel side identification (bit FSDEF = 1 in the SAI_xFRCR register), the tristate mode is managed according to Figure 768 (where bit TRIS in the SAI_xCR1 register = 1, and FSDEF=1, and half frame length is higher than number of slots/2, and NBSLOT=6).
If the TRIS bit in the SAI_xCR2 register is cleared, all the High impedance states on the SD output line on Figure 767 and Figure 768 are replaced by a drive with a value of 0.

53.4.14 Error flags

The SAI implements the following error flags:

- FIFO overrun/underrun
- Anticipated frame synchronization detection
- Late frame synchronization detection
- Codec not ready (AC’97 exclusively)
- Wrong clock configuration in master mode.

FIFO overrun/underrun (OVRUDR)

The FIFO overrun/underrun bit is called OVRUDR in the SAI_xSR register.

The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

Overrun

When the audio block is configured as receiver, an overrun condition may appear if data are received in an audio frame when the FIFO is full and not able to store the received data. In this case, the received data are lost, the flag OVRUDR in the SAI_xSR register is set and an interrupt is generated if OVRUDRIE bit is set in the SAI_xIM register. The slot number, from which the overrun occurs, is stored internally. No more data are stored into the FIFO until it becomes free to store new data. When the FIFO has at least one data free, the SAI audio block receiver stores new data (from new audio frame) from the slot number which was stored internally when the overrun condition was detected. This avoids data slot de-alignment in the destination memory (refer to Figure 769).

The OVRUDR flag is cleared when COVRUDR bit is set in the SAI_xCLRFR register.
Underrun

An underrun may occur when the audio block in the SAI is a transmitter and the FIFO is empty when data need to be transmitted. If an underrun is detected, the slot number for which the event occurs is stored and MUTE value (00) is sent until the FIFO is ready to transmit the data corresponding to the slot for which the underrun was detected (refer to Figure 770). This avoids desynchronization between the memory pointer and the slot in the audio frame.

The underrun event sets the OVRUDR flag in the SAI_xSR register and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. To clear this flag, set COVRUDR bit in the SAI_xCLRFR register.

The underrun event can occur when the audio subblock is configured as master or slave.
**Anticipated frame synchronization detection (AFSDET)**

The AFSDET flag is used only in slave mode. It is never asserted in master mode. It indicates that a frame synchronization (FS) has been detected earlier than expected since the frame length, the frame polarity, the frame offset are defined and known.

Anticipated frame detection sets the AFSDET flag in the SAI_xSR register.

This detection has no effect on the current audio frame which is not sensitive to the anticipated FS. This means that “parasitic” events on signal FS are flagged without any perturbation of the current audio frame.

An interrupt is generated if the AFSDETIE bit is set in the SAI_xIM register. To clear the AFSDET flag, CAFSDET bit must be set in the SAI_xCLRFR register.

To resynchronize with the master after an anticipated frame detection error, four steps are required:

1. Disable the SAI block by resetting SAIEN bit in SAI_xCR1 register. To make sure the SAI is disabled, read back the SAIEN bit and check it is set to 0.
2. Flush the FIFO via FFLUS bit in SAI_xCR2 register.
3. Enable again the SAI peripheral (SAIEN bit set to 1).
4. The SAI block waits for the assertion on FS to restart the synchronization with master.

*Note:* The AFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave. It has no meaning in SPDIF mode since the FS signal is not used.

**Late frame synchronization detection**

The LFSDET flag in the SAI_xSR register can be set only when the SAI audio block operates as a slave. The frame length, the frame polarity and the frame offset configuration are known in register SAI_xFRCR.

If the external master does not send the FS signal at the expecting time thus generating the signal too late, the LFSDET flag is set and an interrupt is generated if LFSDETIE bit is set in the SAI_xIM register.

The LFSDET flag is cleared when CLFSDET bit is set in the SAI_xCLRFR register.

The late frame synchronization detection flag is set when the corresponding error is detected. The SAI needs to be resynchronized with the master (see sequence described in Anticipated frame synchronization detection (AFSDET)).

In a noisy environment, glitches on the SCK clock may be wrongly detected by the audio block state machine and shift the SAI data at a wrong frame position. This event can be detected by the SAI and reported as a late frame synchronization detection error.

There is no corruption if the external master is not managing the audio data frame transfer in continuous mode, which must not be the case in most applications. In this case, the LFSDET flag is set.

*Note:* The LFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave. It has no meaning in SPDIF mode since the signal FS is not used by the protocol.
Codec not ready (CNRDY AC'97)

The CNRDY flag in the SAI_xSR register is relevant only if the SAI audio block is configured to operate in AC'97 mode (PRTCFG[1:0] = 10 in the SAI_xCR1 register). If CNRDYIE bit is set in the SAI_xIM register, an interrupt is generated when the CNRDY flag is set.

CNRDY is asserted when the Codec is not ready to communicate during the reception of the TAG 0 (slot0) of the AC'97 audio frame. In this case, no data are automatically stored into the FIFO since the Codec is not ready, until the TAG 0 indicates that the Codec is ready. All the active slots defined in the SAI_xSLOTR register are captured when the Codec is ready.

To clear CNRDY flag, CCNRDY bit must be set in the SAI_xCLRFR register.

Wrong clock configuration in master mode (with NODIV = 0)

When the audio block operates as a master (MODE[1] = 0) and NODIV bit is equal to 0, the WCKCFG flag is set as soon as the SAI is enabled if the following conditions are met:

- \((FRL+1)\) is not a power of 2, and
- \((FRL+1)\) is not between 8 and 256.

MODE, NODIV, and SAIEN bits belong to SAI_xCR1 register and FRL to SAI_xFRCR register.

If WCKCFGIE bit is set, an interrupt is generated when WCKCFG flag is set in the SAI_xSR register. To clear this flag, set CWCKCFG bit in the SAI_xCLRFR register.

When WCKCFG bit is set, the audio block is automatically disabled, thus performing a hardware clear of SAIEN bit.

53.4.15 Disabling the SAI

The SAI audio block can be disabled at any moment by clearing SAIEN bit in the SAI_xCR1 register. All the already started frames are automatically completed before the SAI stops working. SAIEN bit remains High until the SAI is completely switched-off at the end of the current audio frame transfer.

If an audio block in the SAI operates synchronously with the other one, the one which is the master must be disabled first.

53.4.16 SAI DMA interface

To free the CPU and to optimize bus bandwidth, each SAI audio block has an independent DMA interface to read/write from/to the SAI_xDR register (to access the internal FIFO). There is one DMA channel per audio subblock supporting basic DMA request/acknowledge protocol.

To configure the audio subblock for DMA transfer, set DMAEN bit in the SAI_xCR1 register. The DMA request is managed directly by the FIFO controller depending on the FIFO threshold level (for more details refer to Section 53.4.9: Internal FIFOs). DMA transfer direction is linked to the SAI audio subblock configuration:

- If the audio block operates as a transmitter, the audio block FIFO controller outputs a DMA request to load the FIFO with data written in the SAI_xDR register.
- If the audio block is operated as a receiver, the DMA request is related to read operations from the SAI_xDR register.
Follow the sequence below to configure the SAI interface in DMA mode:
1. Configure SAI and FIFO threshold levels to specify when the DMA request is launched.
2. Configure SAI DMA channel.
3. Enable the DMA.
4. Enable the SAI interface.

*Note:* Before configuring the SAI block, the SAI DMA channel must be disabled.

### 53.5 SAI interrupts

The SAI supports 7 interrupt sources as shown in *Table 574*.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt source</th>
<th>Interrupt group</th>
<th>Audio block mode</th>
<th>Interrupt enable</th>
<th>Interrupt clear</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREQ</td>
<td>FREQ</td>
<td>Master or slave</td>
<td>Receiver or</td>
<td>FREQIE in</td>
<td>Depends on:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receiver or</td>
<td>transmitter</td>
<td>SAI_xIM register</td>
<td>– FIFO threshold setting (FLVL bits</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>in SAI_xCR2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– Communication direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(transmitter or receiver)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>For more details refer to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Section 53.4.9: Internal FIFOs</td>
</tr>
<tr>
<td>OVRUDR</td>
<td>ERROR</td>
<td>Master or slave</td>
<td>Receiver or</td>
<td>OVRUDRIE in</td>
<td>COVRUDR = 1 in SAI_xCLRFR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receiver or</td>
<td>transmitter</td>
<td>SAI_xIM register</td>
<td>register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AFSDET</td>
<td>ERROR</td>
<td>Slave</td>
<td>(not used in AC’97 mode and SPDIF mode)</td>
<td>AFSDETIE in SAI_xIM register</td>
<td>CAFSDET = 1 in SAI_xCLRFR register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LFSDET</td>
<td>ERROR</td>
<td>Slave</td>
<td>(not used in AC’97 mode and SPDIF mode)</td>
<td>LFSDETIE in SAI_xIM register</td>
<td>CLFSDET = 1 in SAI_xCLRFR register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNRDY</td>
<td>ERROR</td>
<td>Slave</td>
<td>(only in AC’97 mode)</td>
<td>CNRDYIE in SAI_xIM register</td>
<td>CCNRDY = 1 in SAI_xCLRFR register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUTEDET</td>
<td>MUTE</td>
<td>Master or slave</td>
<td>Receiver mode only</td>
<td>MUTEDETIE in SAI_xIM register</td>
<td>CMUTEDET = 1 in SAI_xCLRFR register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WCKCFG</td>
<td>ERROR</td>
<td>Master with NODIV = 0 in SAI_xCR1 register</td>
<td>WCKCFGIE in SAI_xIM register</td>
<td>WCKCFG = 1 in SAI_xCLRFR register</td>
<td></td>
</tr>
</tbody>
</table>

*Table 574. SAI interrupt sources*
Follow the sequence below to enable an interrupt:
1. Disable SAI interrupt.
2. Configure SAI.
3. Configure SAI interrupt source.
4. Enable SAI.

53.6 SAI registers

The peripheral registers have to be accessed by words (32 bits).

53.6.1 SAI global configuration register (SAI_GCR)

Address offset: 0x00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SYNOUT[1:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 31:6 Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 5:4 SYNOUT[1:0]: Synchronization outputs

These bits are set and cleared by software.
00: No synchronization output signals. SYNOUT[1:0] must be configured as “No synchronization output signals” when audio block is configured as SPDIF
01: Block A used for further synchronization for others SAI
10: Block B used for further synchronization for others SAI
11: Reserved. These bits must be set when both audio blocks (A and B) are disabled.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 SYNadin[1:0]: Synchronization inputs

These bits are set and cleared by software.
Refer to Table 566: External synchronization selection for information on how to program this field.
These bits must be set when both audio blocks (A and B) are disabled.
They are meaningful if one of the two audio blocks is defined to operate in synchronous mode with an external SAI (SYNcen[1:0] = 10 in SAI_ACR1 or in SAI_BCR1 registers).
53.6.2 SAI configuration register 1 (SAI_ACR1)

Address offset: 0x004
Reset value: 0x0000 0040

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:28: Reserved, must be kept at reset value.

Bit 27 **MCK EN**: Master clock generation enable
- 0: The master clock is not generated
- 1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR**: Oversampling ratio for master clock
- This bit is meaningful only when NODIV bit is set to 0.
- 0: Master clock frequency = $F_{FS} \times 256$
- 1: Master clock frequency = $F_{FS} \times 512$

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
- These bits are set and cleared by software.
- 000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
- Otherwise, the master clock frequency is calculated according to the formula given in Section 53.4.8: SAI clock generator.
- These bits have no meaning when the audio block is slave.
- They have to be configured when the audio block is disabled.

Bit 19 **NODIV**: No divider
- This bit is set and cleared by software.
- 0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
- 1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18: Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
- This bit is set and cleared by software.
- 0: DMA disabled
- 1: DMA enabled

*Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.*
Bit 16  **SAIEN**: Audio block enable  
This bit is set by software.  
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.  
This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.  
0: SAI audio block disabled  
1: SAI audio block enabled.  
*Note*:  When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14  Reserved, must be kept at reset value.

Bit 13  **OUTDRIV**: Output drive  
This bit is set and cleared by software.  
0: Audio block output driven when SAIEN is set  
1: Audio block output driven immediately after the setting of this bit.  
*Note*:  This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12  **MONO**: Mono mode  
This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2.  
When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to [Section : Mono/stereo mode](#) for more details.  
0: Stereo mode  
1: Mono mode.

Bits 11:10  **SYNCEN[1:0]**: Synchronization enable  
These bits are set and cleared by software. They must be configured when the audio subblock is disabled.  
00: audio subblock in asynchronous mode.  
01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode  
10: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock must be configured in Slave mode.  
11: Reserved  
*Note*:  The audio subblock must be configured as asynchronous when SPDIF mode is enabled.

Bit 9  **CKSTR**: Clock strobing edge  
This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.  
0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.  
1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8  **LSBFIRST**: Least significant bit first  
This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC’97 audio protocol since AC’97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.  
0: Data are transferred with MSB first  
1: Data are transferred with LSB first.
Bits 7:5 **DS[2:0]: Data size**

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

- 000: Reserved
- 001: Reserved
- 010: 8 bits
- 011: 10 bits
- 100: 16 bits
- 101: 20 bits
- 110: 24 bits
- 111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]: Protocol configuration**

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

- 00: Free protocol. Free protocol enables to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.
- 01: SPDIF protocol
- 10: AC’97 protocol
- 11: Reserved

Bits 1:0 **MODE[1:0]: SAIx audio block mode**

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

- 00: Master transmitter
- 01: Master receiver
- 10: Slave transmitter
- 11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00).

### 53.6.3 **SAI configuration register 1 (SAI_BCR1)**

Address offset: 0x024

Reset value: 0x0000 0040

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>
Serial audio interface (SAI) RM0481

Bits 31:28  Reserved, must be kept at reset value.

Bit 27 **MCKEN**: Master clock generation enable
0: The master clock is not generated
1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR**: Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
0: Master clock frequency = \( F_{\text{FS}} \times 256 \)
1: Master clock frequency = \( F_{\text{FS}} \times 512 \)

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, The master clock frequency is calculated according to the formula given in Section 53.4.8: SAI clock generator.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19 **NODIV**: No divider
This bit is set and cleared by software.
0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
This bit is set and cleared by software.
0: DMA disabled
1: DMA enabled

*Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.*

Bit 16 **SAIEN**: Audio block enable
This bit is set by software.
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.
This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
0: SAI audio block disabled
1: SAI audio block enabled.

*Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.*

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive
This bit is set and cleared by software.
0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.

*Note: This bit has to be set before enabling the audio block and after the audio block configuration.*
Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to Section : Mono/stereo mode for more details.

0: Stereo mode
1: Mono mode.

Bits 11:10 **SYNCEN[1:0]**: Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.
01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode.
10: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock must be configured in Slave mode.
11: Reserved

*Note: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.*

Bit 9 **CKSTR**: Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.
1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST**: Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC’97 audio protocol since AC’97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first
1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]**: Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved
001: Reserved
010: 8 bits
011: 10 bits
100: 16 bits
101: 20 bits
110: 24 bits
111: 32 bits
53.6.4 SAI configuration register 2 (SAI_ACR2)

Address offset: 0x008
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31-16: Reserved, must be kept at reset value.

Bits 15:14: **COMP[1:0]:** Compressing mode.
   
   These bits are set and cleared by software. The μ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on CPL bit.
   
   The data expansion or data compression are determined by the state of bit MODE[0].
   
   The data compression is applied if the audio block is configured as a transmitter.
   
   The data expansion is automatically applied when the audio block is configured as a receiver.
   
   Refer to Section: Compressing mode for more details.
   
   00: No compressing algorithm
   
   01: Reserved.
   
   10: μ-Law algorithm
   
   11: A-Law algorithm

   Note: Compressing mode is applicable only when Free protocol mode is selected.

Bit 13: **CPL**: Complement bit.
   
   This bit is set and cleared by software.
   
   It defines the type of complement to be used for compressing mode
   
   0: 1’s complement representation.
   
   1: 2’s complement representation.

   Note: This bit has effect only when the compressing mode is μ-Law algorithm or A-Law algorithm.

Bits 12:7: **MUTE[5:0]**: Mute counter.
   
   These bits are set and cleared by software. They are used only in reception mode.
   
   The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.
   
   Refer to Section: Mute mode for more details.

Bit 6: **MUTEVAL**: Mute value.
   
   This bit is set and cleared by software. It must be written before enabling the audio block: SAIEN.
   
   This bit is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.
   
   If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.
   
   If the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.
   
   Refer to Section: Mute mode for more details.
   
   0: Bit value 0 is sent during the mute mode.
   
   1: Last values are sent during the mute mode.

   Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5: **MUTE**: Mute.
   
   This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.
   
   Refer to Section: Mute mode for more details.
   
   0: No mute mode.
   
   1: Mute mode enabled.

   Note: This bit is meaningless and must not be used for SPDIF audio blocks.
Bit 4 TRIS: Tristate management on data line.
This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled. Refer to Section: Output data line management on an inactive slot for more details.
0: SD output line is still driven by the SAI when a slot is inactive.
1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 FFLUSH: FIFO flush.
This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.
0: No FIFO flush.
1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled.

Bits 2:0 FTH[2:0]: FIFO threshold.
This bit is set and cleared by software.
000: FIFO empty
001: ¼ FIFO
010: ½ FIFO
011: ¾ FIFO
100: FIFO full
101: Reserved
110: Reserved
111: Reserved

### 53.6.5 SAI configuration register 2 (SAI_BCR2)

Address offset: 0x028
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-16</td>
<td>Reserved</td>
</tr>
<tr>
<td>15-10</td>
<td>COMP[1:0]</td>
</tr>
<tr>
<td>9-8</td>
<td>CPL</td>
</tr>
<tr>
<td>7-6</td>
<td>MUTECNT[5:0]</td>
</tr>
<tr>
<td>5-4</td>
<td>MUTE</td>
</tr>
<tr>
<td>3-2</td>
<td>TRIS</td>
</tr>
<tr>
<td>1-0</td>
<td>F_FLUSH, FTH[2:0]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Read/Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-10</td>
<td>rw</td>
</tr>
<tr>
<td>9-8</td>
<td>rw</td>
</tr>
<tr>
<td>7-6</td>
<td>rw</td>
</tr>
<tr>
<td>5-4</td>
<td>rw</td>
</tr>
<tr>
<td>3-2</td>
<td>rw</td>
</tr>
<tr>
<td>1-0</td>
<td>rw</td>
</tr>
</tbody>
</table>

2494/3119 RM0481 Rev 1
Bits 31:16  Reserved, must be kept at reset value.

Bits 15:14  **COMP[1:0]**: Comping mode.
These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on **CPL bit**.
The data expansion or data compression are determined by the state of bit **MODE[0]**.
The data compression is applied if the audio block is configured as a transmitter.
The data expansion is automatically applied when the audio block is configured as a receiver.
Refer to **Section : Comping mode** for more details.
00: No companding algorithm
01: Reserved.
10: µ-Law algorithm
11: A-Law algorithm

*Note:* **Comping mode is applicable only when Free protocol mode is selected.**

Bit 13  **CPL**: Complement bit.
This bit is set and cleared by software.
It defines the type of complement to be used for companding mode
0: 1’s complement representation.
1: 2’s complement representation.

*Note:* **This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.**

Bits 12:7  **MUTECNT[5:0]**: Mute counter.
These bits are set and cleared by software. They are used only in reception mode.
The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.
Refer to **Section : Mute mode** for more details.

Bit 6  **MUTEVAL**: Mute value.
This bit is set and cleared by software. It must be written before enabling the audio block: SAIEN.
This bit is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.
If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.
if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.
Refer to **Section : Mute mode** for more details.
0: Bit value 0 is sent during the mute mode.
1: Last values are sent during the mute mode.

*Note:* **This bit is meaningless and must not be used for SPDIF audio blocks.**

Bit 5  **MUTE**: Mute.
This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.
Refer to **Section : Mute mode** for more details.
0: No mute mode.
1: Mute mode enabled.

*Note:* **This bit is meaningless and must not be used for SPDIF audio blocks.**
53.6.6 SAI frame configuration register (SAI_AFRCR)

Address offset: 0x00C
Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSOFF</td>
<td>FSPOL</td>
<td>FSDEF</td>
<td>FSOFF</td>
<td>FSPOL</td>
<td>FSDEF</td>
<td>FSOFF</td>
<td>FSPOL</td>
<td>FSDEF</td>
<td>FSOFF</td>
<td>FSPOL</td>
<td>FSDEF</td>
<td>FSOFF</td>
<td>FSPOL</td>
<td>FSDEF</td>
<td>FSOFF</td>
<td>FSPOL</td>
<td>FSDEF</td>
<td>FSOFF</td>
<td>FSPOL</td>
<td>FSDEF</td>
<td>FSOFF</td>
<td>FSPOL</td>
<td>FSDEF</td>
<td>FSOFF</td>
<td>FSPOL</td>
<td>FSDEF</td>
<td>FSOFF</td>
<td>FSPOL</td>
<td>FSDEF</td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bit 4 TRIS: Tristate management on data line.
This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.
Refer to Section: Output data line management on an inactive slot for more details.
0: SD output line is still driven by the SAI when a slot is inactive.
1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 FFLUSH: FIFO flush.
This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.
0: No FIFO flush.
1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled.

Bits 2:0 FTH[2:0]: FIFO threshold.
This bit is set and cleared by software.
000: FIFO empty
001: ¼ FIFO
010: ½ FIFO
011: ¾ FIFO
100: FIFO full
101: Reserved
110: Reserved
111: Reserved
Bits 31:19  Reserved, must be kept at reset value.

Bit 18  **FSOFF**: Frame synchronization offset.
   This bit is set and cleared by software. It is meaningless and is not used in AC’97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
   0: FS is asserted on the first bit of the slot 0.
   1: FS is asserted one bit before the first bit of the slot 0.

Bit 17  **FSPOL**: Frame synchronization polarity.
   This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC’97 or SPDIF audio block configuration.
   This bit must be configured when the audio block is disabled.
   0: FS is active low (falling edge)
   1: FS is active high (rising edge)

Bit 16  **FSDEF**: Frame synchronization definition.
   This bit is set and cleared by software.
   0: FS signal is a start frame signal
   1: FS signal is a start of frame signal + channel side identification
   When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots are dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
   This bit is meaningless and is not used in AC’97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15  Reserved, must be kept at reset value.

Bits 14:8  **FSALL[6:0]**: Frame synchronization active level length.
   These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame.
   These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration.
   They must be configured when the audio block is disabled.

Bits 7:0  **FRL[7:0]**: Frame length.
   These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.
   The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).
   In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.
   These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration. They must be configured when the audio block is disabled.
53.6.7 **SAI frame configuration register (SAI_BFRCR)**

Address offset: 0x02C  
Reset value: 0x0000 0007

**Note:** This register has no meaning in AC’97 and SPDIF audio protocol

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res.</td>
<td>FSALL[6:0]</td>
<td>FRL[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:19 Reserved, must be kept at reset value.

**Bit 18 FSOFF:** Frame synchronization offset.  
This bit is set and cleared by software. It is meaningless and is not used in AC’97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.  
0: FS is asserted on the first bit of the slot 0.  
1: FS is asserted one bit before the first bit of the slot 0.

**Bit 17 FSPOL:** Frame synchronization polarity.  
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC’97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.  
0: FS is active low (falling edge)  
1: FS is active high (rising edge)

**Bit 16 FSDEF:** Frame synchronization definition.  
This bit is set and cleared by software.  
0: FS signal is a start frame signal  
1: FS signal is a start of frame signal + channel side identification

When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots is dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols…).  
This bit is meaningless and is not used in AC’97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.
Bit 15  Reserved, must be kept at reset value.

Bits 14:8  **FSALL[6:0]**: Frame synchronization active level length.
These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame.
These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.
They must be configured when the audio block is disabled.

Bits 7:0  **FRL[7:0]**: Frame length.
These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.
The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).
In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.
These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

### 53.6.8  SAI slot register (SAI_ASLOTR)

Address offset: 0x010
Reset value: 0x0000 0000

**Note:** This register has no meaning in AC'97 and SPDIF audio protocol.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16  **SLOTEN[15:0]**: Slot enable.
These bits are set and cleared by software.
Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
0: Inactive slot.
1: Active slot.
The slot must be enabled when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

Bits 15:12  Reserved, must be kept at reset value.

Bits 11:8  **NBSLOT[3:0]**: Number of slots in an audio frame.
These bits are set and cleared by software.
The value set in this bitfield represents the number of slots + 1 in the audio frame (including the number of inactive slots). The maximum number of slots is 16.
The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.
The number of slots must be configured when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
Bits 7:6 **SLOTSZ[1:0]:** Slot size

This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC’97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]:** First bit offset

These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC’97 or SPDIF mode.

### 53.6.9 SAI slot register (SAI_BSLOTR)

Address offset: 0x030
Reset value: 0x0000 0000

*Note:* This register has no meaning in AC’97 and SPDIF audio protocol.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Bits 31:16 **SLOTEN[15:0]**: Slot enable.

These bits are set and cleared by software.

Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).

- 0: Inactive slot.
- 1: Active slot.

The slot must be enabled when the audio block is disabled.

They are ignored in AC’97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.

These bits are set and cleared by software.

The value set in this bitfield represents the number of slots + 1 in the audio frame (including the number of inactive slots). The maximum number of slots is 16.

The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.

The number of slots must be configured when the audio block is disabled.

They are ignored in AC’97 or SPDIF mode.

Bits 7:6 **SLOTSZ[1:0]**: Slot size

This bits is set and cleared by software.

The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.

Refer to *Output data line management on an inactive slot* for information on how to drive SD line.

These bits must be set when the audio block is disabled.

They are ignored in AC’97 or SPDIF mode.

- 00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
- 01: 16-bit
- 10: 32-bit
- 11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset

These bits are set and cleared by software.

The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.

These bits must be set when the audio block is disabled.

They are ignored in AC’97 or SPDIF mode.

### 53.6.10 SAI interrupt mask register (SAI_AIM)

Address offset: 0x014

Reset value: 0x0000 0000
Bits 31:7  Reserved, must be kept at reset value.

Bit 6  **LFSDETIE**: Late frame synchronization detection interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
   This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 5  **AFSDDETIE**: Anticipated frame synchronization detection interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When this bit is set, an interrupt is generated if the AFSDDET bit in the SAI_xSR register is set.
   This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 4  **CNRDYIE**: Codec not ready interrupt enable (AC’97).
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC’97 frame if the
   Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR
   register is set and an interrupt is generated.
   This bit has a meaning only if the AC’97 mode is selected through PRTCFG[1:0] bits and the audio
   block is operates as a receiver.

Bit 3  **FREQIE**: FIFO request interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
   Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be
   configured before setting FREQIE to avoid a parasitic interrupt in receiver mode.

Bit 2  **WCKCFGIE**: Wrong clock configuration interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and
   NODIV = 0.
   It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
   *Note*: This bit is used only in Free protocol mode and is meaningless in other modes.

Bit 1  **MUTEDETIE**: Mute detection interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
   This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0  **OVRUDRIE**: Overrun/underrun interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.
53.6.11 SAI interrupt mask register (SAI_BIM)

Address offset: 0x034
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>IE</td>
<td>IE</td>
<td>IE</td>
<td>IE</td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

- **Bit 6 LFSDETIE**: Late frame synchronization detection interrupt enable.
  - This bit is set and cleared by software.
  - 0: Interrupt is disabled
  - 1: Interrupt is enabled
  - When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
  - This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

- **Bit 5 AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
  - This bit is set and cleared by software.
  - 0: Interrupt is disabled
  - 1: Interrupt is enabled
  - When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
  - This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

- **Bit 4 CNRDYIE**: Codec not ready interrupt enable (AC'97).
  - This bit is set and cleared by software.
  - 0: Interrupt is disabled
  - 1: Interrupt is enabled
  - When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC’97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
  - This bit has a meaning only if the AC’97 mode is selected through PRTCFG[1:0] bits and the audio block operates as a receiver.

- **Bit 3 FREQIE**: FIFO request interrupt enable.
  - This bit is set and cleared by software.
  - 0: Interrupt is disabled
  - 1: Interrupt is enabled
  - When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
  - Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,
Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
- This bit is set and cleared by software.
- 0: Interrupt is disabled
- 1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
*Note: This bit is used only in Free protocol mode and is meaningless in other modes.*

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
- This bit is set and cleared by software.
- 0: Interrupt is disabled
- 1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
- This bit is set and cleared by software.
- 0: Interrupt is disabled
- 1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

### 53.6.12 SAI status register (SAI_ASR)

Address offset: 0x018
Reset value: 0x0000 0008

| Bit 31:19 | Reserved, must be kept at reset value. |
| Bit 18:16 | **FLVL[2:0]**: FIFO level threshold. |
|           | This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode). |
|           | 000: FIFO empty (transmitter and receiver modes) |
|           | 001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode) |
|           | 010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode) |
|           | 011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode) |
|           | 100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode) |
|           | 101: FIFO full (transmitter and receiver modes) |
|           | Others: Reserved |
| Bit 15:7  | Reserved, must be kept at reset value. |
Bit 6 **LFSDET**: Late frame synchronization detection.
   This bit is read only.
   0: No error.
   1: Frame synchronization signal is not present at the right time.
   This flag can be set only if the audio block is configured in slave mode.
   It is not used in AC’97 or SPDIF mode.
   It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
   This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register.

Bit 5 **AFSDET**: Anticipated frame synchronization detection.
   This bit is read only.
   0: No error.
   1: Frame synchronization signal is detected earlier than expected.
   This flag can be set only if the audio block is configured in slave mode.
   It is not used in AC’97 or SPDIF mode.
   It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
   This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.
   This bit is read only.
   0: External AC’97 Codec is ready
   1: External AC’97 Codec is not ready
   This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register and
   configured in receiver mode.
   It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
   This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
   This bit is read only.
   0: No FIFO request.
   1: FIFO request to read or to write the SAI_xDR.
   The request depends on the audio block configuration:
   – If the block is configured in transmission mode, the FIFO request is related to a write request
     operation in the SAI_xDR.
   – If the block configured in reception, the FIFO request related to a read request operation from the
     SAI_xDR.
   This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.
Bit 2 **WCKCFG**: Wrong clock configuration flag.
   This bit is read only.
   0: Clock configuration is correct
   1: Clock configuration does not respect the rule concerning the frame length specification defined in
   Section 53.4.6: Frame synchronization (configuration of FRL[7:0] bit in the SAI_xFRCR register)
   This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
   It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
   This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
   This bit is read only.
   0: No MUTE detection on the SD input line
   1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio
   frame
   This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a
   consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
   It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
   This flag is cleared when the software sets CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
   This bit is read only.
   0: No overrun/underrun error.
   1: Overrun/underrun error detection.
   The overrun and underrun conditions can occur only when the audio block is configured as a
   receiver and a transmitter, respectively.
   It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
   This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

### 53.6.13 SAI status register (SAI_BSR)

Address offset: 0x038

Reset value: 0x0000 0008

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.
This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC’97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register.

Bit 5 **AFSDET**: Anticipated frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC’97 or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.
This bit is read only.
0: External AC’97 Codec is ready
1: External AC’97 Codec is not ready
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:
– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.
This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.
Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in Section 53.4.6: Frame synchronization (configuration of FRL[7:0] bit in the SAI_xFRCR register)
This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
This bit is read only.
0: No MUTE detection on the SD input line
1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frames
This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
This bit is read only.
0: No overrun/underrun error.
1: Overrun/underrun error detection.
The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.
It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

### 53.6.14 SAI clear flag register (SAI_ACLRFR)
Address offset: 0x01C
Reset value: 0x0000 0000
Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

### 53.6.15 SAI clear flag register (SAI_BCLRFR)

Address offset: 0x03C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

RM0481 Rev 1 2509/3119
Bits 31:7  Reserved, must be kept at reset value.

Bit 6  **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 5  **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4  **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3  Reserved, must be kept at reset value.

Bit 2  **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1  **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0  **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

### 53.6.16  SAI data register (SAI_ADR)

Address offset: 0x020

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31:16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15:0</th>
</tr>
</thead>
</table>

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 2510/3119  | RM0481 Rev 1 |
53.6.17 SAI data register (SAI_BDR)

Address offset: 0x040
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 **DATA[31:0]: Data**
- A write to this register loads the FIFO provided the FIFO is not full.
- A read from this register empties the FIFO if the FIFO is not empty.

53.6.18 SAI PDM control register (SAI_PDMCR)

Address offset: 0x0044
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.
Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CKEN2:** Clock enable of bitstream clock number 2
- This bit is set and cleared by software.
- 0: SAI_CK2 clock disabled
- 1: SAI_CK2 clock enabled

Note: **It is not recommended to configure this bit when PDMEN = 1.**
- **SAI_CK2 might not be available for all SAI instances. Refer to Section 53.3: SAI implementation for details.**
Bit 8 **CKEN1**: Clock enable of bitstream clock number 1
This bit is set and cleared by software.
0: SAI_CK1 clock disabled
1: SAI_CK1 clock enabled

*Note*: It is not recommended to configure this bit when PDMEN = 1.
SAI_CK1 might not be available for all SAI instances. Refer to Section 53.3: SAI implementation for details.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MICNBR[1:0]**: Number of microphones
This bit is set and cleared by software.
00: Configuration with 2 microphones
01: Configuration with 4 microphones
10: Configuration with 6 microphones
11: Configuration with 8 microphones

*Note*: It is not recommended to configure this field when PDMEN = 1.*
The complete set of data lines might not be available for all SAI instances. Refer to Section 53.3: SAI implementation for details.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PDMEN**: PDM enable
This bit is set and cleared by software. This bit enables to control the state of the PDM interface block.
Make sure that the SAI in already operating in TDM master mode before enabling the PDM interface.
0: PDM interface disabled
1: PDM interface enabled

### 53.6.19 SAI PDM delay register (SAI_PDMDLY)

Address offset: 0x0048
Reset value: 0x0000 0000
Bit 31  Reserved, must be kept at reset value.

Bits 30:28  **DLYM4R[2:0]**: Delay line for second microphone of **pair 4**

  This bit is set and cleared by software.
  000: No delay
  001: Delay of 1 $T_{SAI\_CK}$ period
  010: Delay of 2 $T_{SAI\_CK}$ periods
  ...
  111: Delay of 7 $T_{SAI\_CK}$ periods

  This field can be changed on-the-fly.
  Note:  **This field can be used only if D4 line is available.** Refer to Section 53.3: SAI implementation to check if it is available.

Bit 27  Reserved, must be kept at reset value.

Bits 26:24  **DLYM4L[2:0]**: Delay line for first microphone of pair 4

  This bit is set and cleared by software.
  000: No delay
  001: Delay of 1 $T_{SAI\_CK}$ period
  010: Delay of 2 $T_{SAI\_CK}$ periods
  ...
  111: Delay of 7 of $T_{SAI\_CK}$ periods

  This field can be changed on-the-fly.
  Note:  **This field can be used only if D4 line is available.** Refer to Section 53.3: SAI implementation to check if it is available.

Bit 23  Reserved, must be kept at reset value.

Bits 22:20  **DLYM3R[2:0]**: Delay line for second microphone of pair 3

  This bit is set and cleared by software.
  000: No delay
  001: Delay of 1 $T_{SAI\_CK}$ period
  010: Delay of 2 $T_{SAI\_CK}$ periods
  ...
  111: Delay of 7 $T_{SAI\_CK}$ periods

  This field can be changed on-the-fly.
  Note:  **This field can be used only if D3 line is available.** Refer to Section 53.3: SAI implementation to check if it is available.

Bit 19  Reserved, must be kept at reset value.

Bits 18:16  **DLYM3L[2:0]**: Delay line for first microphone of pair 3

  This bit is set and cleared by software.
  000: No delay
  001: Delay of 1 $T_{SAI\_CK}$ period
  010: Delay of 2 $T_{SAI\_CK}$ periods
  ...
  111: Delay of 7 $T_{SAI\_CK}$ periods

  This field can be changed on-the-fly.
  Note:  **This field can be used only if D3 line is available.** Refer to Section 53.3: SAI implementation to check if it is available.
Serial audio interface (SAI)

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DLYM2R[2:0]**: Delay line for second microphone of pair 2
   This bit is set and cleared by software.
   000: No delay
   001: Delay of 1 $T_{SAI\_CK}$ period
   010: Delay of 2 $T_{SAI\_CK}$ periods
   ...  
   111: Delay of 7 $T_{SAI\_CK}$ periods

   This field can be changed on-the-fly.
   *Note: This field can be used only if D2 line is available. Refer to Section 53.3: SAI implementation to check if it is available.*

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DLYM2L[2:0]**: Delay line for first microphone of pair 2
   This bit is set and cleared by software.
   000: No delay
   001: Delay of 1 $T_{SAI\_CK}$ period
   010: Delay of 2 $T_{SAI\_CK}$ periods
   ...  
   111: Delay of 7 $T_{SAI\_CK}$ periods

   This field can be changed on-the-fly.
   *Note: This field can be used only if D2 line is available. Refer to Section 53.3: SAI implementation to check if it is available.*

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DLYM1R[2:0]**: Delay line adjust for second microphone of pair 1
   This bit is set and cleared by software.
   000: No delay
   001: Delay of 1 $T_{SAI\_CK}$ period
   010: Delay of 2 $T_{SAI\_CK}$ periods
   ...  
   111: Delay of 7 $T_{SAI\_CK}$ periods

   This field can be changed on-the-fly.
   *Note: This field can be used only if D1 line is available. Refer to Section 53.3: SAI implementation to check if it is available.*

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **DLYM1L[2:0]**: Delay line adjust for first microphone of pair 1
   This bit is set and cleared by software.
   000: No delay
   001: Delay of 1 $T_{SAI\_CK}$ period
   010: Delay of 2 $T_{SAI\_CK}$ periods
   ...  
   111: Delay of 7 $T_{SAI\_CK}$ periods

   This field can be changed on-the-fly.
   *Note: This field can be used only if D1 line is available. Refer to Section 53.3: SAI implementation to check if it is available.*
### 53.6.20 SAI register map

#### Table 575. SAI register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Field</th>
<th>Reset value</th>
<th>Value</th>
</tr>
</thead>
</table>
| 0x0000       | SAI_GCR           |             |             | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Table 575. SAI register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>rest value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0048</td>
<td>SAI_PDMDLY</td>
<td>DLYMRR(2:0) DLYML(2:0) DLYMRR(2:0) DLYML(2:0) DLYMRR(2:0) DLYML(2:0) DLYMRR(2:0) DLYML(2:0)</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for the register boundary addresses.
54 FD controller area network (FDCAN)

54.1 Introduction

The controller area network (CAN) subsystem (see Figure 771) consists of one CAN module, a shared message RAM and a configuration block. Refer to the memory map for the base address of each of these parts.

The modules (FDCAN) are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

A 0.8-Kbyte message RAM per FDCAN instance implements filters, receive FIFOs, transmit event FIFOs and transmit FIFOs.

The CAN subsystem I/O signals and pins are detailed, respectively, in Table 576 and Figure 771.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fdcan_ck</td>
<td>Digital input</td>
<td>CAN subsystem kernel clock input</td>
</tr>
<tr>
<td>fdcan_pclk</td>
<td></td>
<td>CAN subsystem APB interface clock input</td>
</tr>
<tr>
<td>fdcan_intr0_it</td>
<td>Digital output</td>
<td>FDCAN interrupt0</td>
</tr>
<tr>
<td>fdcan_intr1_it</td>
<td></td>
<td>FDCAN interrupt1</td>
</tr>
<tr>
<td>fdcan_ts[0:15]</td>
<td>-</td>
<td>External timestamp vector</td>
</tr>
<tr>
<td>FDCAN_RX</td>
<td>Digital input</td>
<td>FDCAN receive pin</td>
</tr>
<tr>
<td>FDCAN_TX</td>
<td>Digital output</td>
<td>FDCAN transmit pin</td>
</tr>
<tr>
<td>APB interface</td>
<td>Digital input/output</td>
<td>Single APP with multiple psel for configuration, control and RAM access</td>
</tr>
</tbody>
</table>
Figure 771. CAN subsystem

- **FD controller area network (FDCAN)**

- **Controller Area Network (CAN)**

- **Controller / Arbiter**

- **Message RAM**
  - Buffers
  - FIFOs
  - Filters

- **RAM APB**

- **Interrupts interface**

- **Message RAM interface**

- **CAN core**

- **TX Handler**
  - TX prioritization
  - Frame Synchronization

- **RX Handler**
  - Acceptance filter

- **Configuration**
  - Sub system Config reg

- **Control and Configuration registers**

- **RAM Controller / Arbiter**

- **APB clock domain**

- **Kernel clock domain**

- **32-bit APB bus**

- **fdcan_clk**

- **fdcan1_intr0_it**

- **fdcan1_intr1_it**

- **fdcan_pclk**

- **fdcan1_ts[0:15]**

- **FD CAN TX1**

- **FD CAN RX1**

- **FD CAN TX2**

- **FD CAN RX2**

- **CFG_APB**

- **RAM_APB**

- **Ctrl_APB**

- **CKDIV**

- **fdcan_tq_ck**

- **fdcan_tq_ck**

- **Sync**

- **Tx Req**

- **Tx State**

- **Rx State**

- **CANFDI**

- **CANFDI**

- **/ 1..30**

- **Kernel clock domain**
54.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B and ISO 11898-1: 2015, -4
- CAN FD with maximum 64 Data bytes supported
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two receive FIFOs of three payloads each (up to 64 bytes per payload)
- Separate signaling on reception of high priority messages
- Configurable Transmit FIFO / queue of three payload (up to 64 bytes per payload)
- Transmit event FIFO
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power down support
54.3 FDCAN functional description

The FDCAN peripheral provides two interrupt lines, `fdcan_intr0_it` and `fdcan_intr1_it`.

By programming EINT0 and EINT1 bits in FDCAN_IIE register, the interrupt lines can be separately enabled or disabled.

**CAN core**

The CAN core contains the protocol controller and receive / transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions and supports both 11-bit and 29-bit identifiers.

**Sync**

The Sync block synchronizes signals from the APB clock domain to the CAN kernel clock domain and vice versa.
**Tx handler**

Controls the message transfer from the Message RAM to the CAN core. A maximum of three Tx buffers is available for transmission. Tx buffer can be used as Tx FIFO or a Tx queue. Tx event FIFO stores Tx timestamps together with the corresponding Message ID. Transmit cancellation is also supported.

**Rx handler**

Controls the transfer of received messages from the CAN core to the external Message RAM. The Rx handler supports two receive FIFOs, for storage of all messages that have passed acceptance filtering. An Rx timestamp is stored together with each message. Up to 28 filters can be defined for 11-bit IDs, up to 8 filters for 29-bit IDs.

**APB interface**

Connects the FDCAN to the APB bus for configuration registers, controller configuration and RAM access.

**Message RAM interface**

Connects the FDCAN access to an external 1-Kbyte Message RAM through a RAM controller / arbiter.

### 54.3.1 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

As shown in Figure 773, its operation may be explained simply by splitting the bit time in three segments, as follows:

- **Synchronization segment (SYNC_SEG):** a bit change is expected to occur within this time segment, having a fixed length of one time quantum (1 x tq).
- **Bit segment 1 (BS1):** defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable between 1 and 16 time quanta, but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of various nodes of the network.
- **Bit segment 2 (BS2):** defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between one and eight time quanta, but may also be automatically shortened to compensate for negative phase drifts.
The baud rate is the inverse of bit time (baud rate = 1 / bit time), which, in turn, is the sum of three components. **Figure 773** indicates that bit time = \( t_{\text{SyncSeg}} + t_{\text{BS1}} + t_{\text{BS2}} \), where:

- for the nominal bit time
  - \( t_q = (\text{FDCAN\_NBTP\_NBRP}[8:0] + 1) \times t_{\text{fc}} \)
  - \( t_{\text{SyncSeg}} = t_q \)
  - \( t_{\text{BS1}} = t_q \times (\text{FDCAN\_NBTP\_NTSEG1}[7:0] + 1) \)
  - \( t_{\text{BS2}} = t_q \times (\text{FDCAN\_NBTP\_NTSEG2}[6:0] + 1) \)

- for the data bit time
  - \( t_q = (\text{FDCAN\_DBTP\_DBRP}[4:0] + 1) \times t_{\text{fc}} \)
  - \( t_{\text{SyncSeg}} = t_q \)
  - \( t_{\text{BS1}} = t_q \times (\text{FDCAN\_DBTP\_DTSEG1}[4:0] + 1) \)
  - \( t_{\text{BS2}} = t_q \times (\text{FDCAN\_DBTP\_DTSEG2}[3:0] + 1) \)

The (Re)Synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the Bit timing register is only possible while the device is in Standby mode. Registers FDCAN\_DBTP and FDCAN\_NBTP (dedicated, respectively, to data and nominal bit timing) are only accessible when CCCR.CCE and CCCR.INIT are set.

**Note:** For a detailed description of the CAN bit timing and resynchronization mechanism refer to the ISO 11898-1 standard.

### 54.3.2 Operating modes

#### Configuration

Access to IP version, hardware and input clock divider configuration. When the clock divider is set to 0, the primary input clock is used as it is.
Software initialization

Software initialization is started by setting INIT bit in FDCAN_CCCR register, either by software or by a hardware reset, or by going Bus_Off. While INIT bit in FDCAN_CCCR register is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output FDCAN_TX is recessive (high). The EML (error management logic) counters are unchanged. Setting INIT bit in FDCAN_CCCR does not change any configuration register. Clearing INIT bit in FDCAN_CCCR finishes the software initialization. Afterwards the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when both INIT bit in FDCAN_CCCR register and CCE bit in FDCAN_CCCR register are set.

CCE bit in FDCAN_CCCR register can only be set/cleared while INIT bit in FDCAN_CCCR is set. CCE bit in FDCAN_CCCR register is automatically cleared when INIT bit in FDCAN_CCCR is cleared.

The following registers are reset when CCE bit in FDCAN_CCCR register is set:

- FDCAN_HPMS - High priority message status
- FDCAN_RXF0S - Rx FIFO 0 status
- FDCAN_RXF1S - Rx FIFO 1 status
- FDCAN_TXFQS - Tx FIFO/Queue status
- FDCAN_TXBRP - Tx buffer request pending
- FDCAN_TXBTO - Tx buffer transmission occurred
- FDCAN_TXBCF - Tx buffer cancellation finished
- FDCAN_TXEFS - Tx event FIFO status

The timeout counter value TOC bit in FDCAN_TOCV register is preset to the value configured by TOP bit in FDCAN_TOCC register when CCE bit in FDCAN_CCCR is set.

In addition the state machines of the Tx Handler and Rx handler are held in idle state while CCE bit in FDCAN_CCCR is set.

The following registers can be written only when CCE bit in FDCAN_CCCR register is cleared:

- TXBAR - Tx buffer add request
- TXBCR - Tx buffer cancellation request

TEST bit in FDCAN_CCCR and MON bit in FDCAN_CCCR can only be set by software while both INIT bit in CCCR and CCE bit in CCCR register are set. Both bits may be reset at any time. DAR bit in FDCAN_CCCR can only be set/cleared while both INIT bit in FDCAN_CCCR and CCE bit in FDCAN_CCCR are set.

Normal operation

The FDCAN default operating mode after hardware reset is event-driven CAN communication. TT Operation Mode is not supported.

Once the FDCAN is initialized and INIT bit in FDCAN_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into the Rx FIFO 0 or Rx FIFO 1.
For messages to be transmitted, Tx FIFO or Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

**CAN FD operation**

There are two variants in the FDCAN protocol:

1. **Long frame mode (LFM)**, where the data field of a CAN frame may be longer than eight bytes
2. **Fast frame mode (FFM)**, where control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate compared to the beginning and to the end of the frame

Fast Frame Mode can be used in combination with Long Frame Mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers are decoded as FDF bit: FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame.

In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside this CAN FD frame is switched. A CAN FD bit rate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for future expansion of the protocol. In case the FDCAN receives a frame with FDF recessive and res recessive, it signals a Protocol exception event by setting bit PSR.PXE. When Protocol exception handling is enabled (CCCR.PXHD = 0), this causes the operation state to change from Receiver (PSR.ACT = 10) to Integrating (PSR.ACT = 00) at the next sample point. In case Protocol exception Handling is disabled (CCCR.PXHD = 1), the FDCAN treats a recessive res bit as a form error and responds with an error frame.

CAN FD operation is enabled by programming CCCR.FDOE. In case CCCR.FDOE = 1, transmission and reception of CAN FD frames is enabled. Transmission and reception of Classic CAN frames is always possible. Whether a CAN FD frame or a classic CAN frame is transmitted can be configured via bit FDF in the respective Tx buffer element. With CCCR.FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit FDF of a Tx buffer element is set. CCCR.FDOE and CCCR.BRSE can only be changed while CCCR.INIT and CCCR.CCE are both set.

With CCCR.FDOE = 0, the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format. With CCCR.FDOE = 1 and CCCR.BRSE = 0, only bit FDF of a Tx buffer element is evaluated. With CCCR.FDOE = 1 and CCCR.BRSE = 1, transmission of CAN FD frames with bit rate switching is enabled. All Tx buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.
A mode change during CAN operation is recommended only under the following conditions:

- The failure rate in the CAN FD data phase is significantly higher than in the CAN FD arbitration phase. In this case, disable the CAN FD bit rate switching option for transmissions.
- During system startup, all nodes are transmitting Classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking have to be transmitted in Classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming is completed. Then all nodes switch back to Classic CAN communication.

In the FDCAN format, the coding of the DLC differs from the one of the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to Table 577.

In CAN FD Fast frames, the bit timing is switched inside the frame, after the BRS (bit rate switch) bit, if this bit is recessive. Before the BRS bit, in the FDCAN arbitration phase, the standard CAN bit timing is used as defined by the Bit timing and prescaler register BTP. In the following FDCAN data phase, the fast CAN bit timing is used as defined by the Fast bit timing and prescaler register FBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with a FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta (tq), the bit rate in the data phase is 5 Mbit/s.

In both data frame formats (CAN FD long frames and CAN FD fast frames), the value of bit ESI (error status indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case a FDCAN Tx buffer is configured for FDCAN transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives a FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx FIFO. The remaining bytes are discarded.

**Transceiver delay compensation**

During the data phase of a FDCAN transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN_TX the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN_RX. The received data is delayed by the CAN transceiver loop delay. If this delay is

<table>
<thead>
<tr>
<th>DLC</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Data bytes</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>32</td>
<td>48</td>
<td>64</td>
</tr>
</tbody>
</table>
greater than TSEG1 (time segment before sample point), a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of a FDCAN frame is limited by the transceivers loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bit rates during the FDCAN data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the secondary sample point (SSP). If a bit error is detected, the transmitter reacts on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit DBTP.TDC.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset as configured by TDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq (minimum time quantum, that is one period of fdcan_tq_ck clock).

PSR.TDCV shows the actual transmitter delay compensation value. PSR.TDCV is cleared when CCCR.INIT is set and is updated at each transmission of an FD frame while DBTP.TDC is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from FDCAN_Tx to FDCAN_Rx and the configured transmitter delay compensation offset TDCR.TDCO has to be lower than 6 bit times in the data phase.
- The sum of the measured delay from FDCAN_TX to FDCAN_RX and the configured transmitter delay compensation offset TDCR.TDCO has to be lower than or equal to 127 mtq. If the sum exceeds this value, the maximum value (127 mtq) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, which stops checking received bits at the SSPs.

If transmitter delay compensation is enabled by programming DBTP.TDC = 1, the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN_TX of the transmitter. The resolution of this measurement is one mtq.
To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit (resulting in a too early SSP position), the use of a transmitter delay compensation filter window can be enabled by programming TDCR.TDCF. This defines a minimum value for the SSP position. Dominant edges on FDCAN_RX, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCR.TDCF and FDCAN_RX is low.

**Restricted operation mode**

In Restricted operation mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters (ECR.REC, ECR.TEC) are frozen while error logging (ECR.CEL) is active. The software can set the FDCAN into Restricted operation mode by setting bit CCCR.ASM. The bit can only be set by software when both CCCR.CCE and CCCR.INIT are set to 1. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted operation mode the software has to reset CCCR.ASM.

The Restricted operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted operation mode after it has received a valid frame.

**Note:** The Restricted operation mode must not be combined with the Loop back mode (internal or external).
Bus monitoring mode

The FDCAN is set in Bus monitoring mode by setting CCCR.MON bit. In Bus monitoring mode (for more details refer to ISO11898-1, 10.12 Bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN can monitor it, even if the CAN bus remains in recessive state. In Bus monitoring mode the TXBRP register is held in reset state.

The Bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. **Figure 775** shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in Bus monitoring mode.

**Figure 775. Pin control in Bus monitoring mode**

Disabled automatic retransmission (DAR) mode

According to the CAN specification (see ISO11898-1, 6.3.3 Recovery Management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled.

Frame transmission in DAR mode

In DAR mode all transmissions are automatically canceled after they have been started on the CAN bus. A Tx buffer Tx Request Pending bit TXBRP.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, or has been aborted due to lost arbitration, or when an error has occurred during frame transmission.
• Successful transmission
  – Corresponding Tx buffer transmission occurred bit TXBTO[TOx] set
  – Corresponding Tx buffer cancellation finished bit TXBCF[CFx] not set
• Successful transmission in spite of cancellation
  – Corresponding Tx buffer transmission occurred bit TXBTO[TOx] set
  – Corresponding Tx buffer cancellation finished bit TXBCF[CFx] set
• Arbitration loss or frame transmission disturbed
  – Corresponding Tx buffer transmission occurred bit TXBTO[TOx] not set
  – Corresponding Tx buffer cancellation finished bit TXBCF[CFx] set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx event FIFO element is written with Event Type ET = 10 (transmission in spite of cancellation).

Power down (Sleep mode)

The FDCAN can be set into power down mode controlled by clock stop request input via CC control register CCCR[CSR]. As long as the clock stop request is active, bit CCCR[CSR] is read as 1.

When all pending transmission requests have completed, the FDCAN waits until bus idle state is detected. Then the FDCAN sets then CCCR[INIT] to 1 to prevent any further CAN transfers. Now the FDCAN acknowledges that it is ready for power down by setting CCCR[CSA] to 1. In this state, before the clocks are switched off, further register accesses can be made. A write access to CCCR[INIT] has no effect. Now the module clock inputs may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting CC control register flag CCCR.CSR. The FDCAN acknowledges this by resetting CCCR[CSA]. Afterwards, the application can restart CAN communication by resetting bit CCCR[INIT].

Test modes

To enable write access to FDCAN test register (FDCAN_TEST), bit CCCR.TEST must be set to 1, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN_TX by programming TEST.TX. In addition to its default function (the serial data output) it can drive the CAN Sample Point signal to monitor the FDCAN bit timing and it can drive constant dominant or recessive values. The actual value at pin FDCAN_RX can be read from TEST.RX. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there may be a delay of several APB clock periods between writing to TEST.TX until the new configuration is visible at FDCAN_TX output pin. This applies also when reading FDCAN_RX input pin via TEST.RX.

Note: Test modes must be used for production tests or self test only. The software control for FDCAN_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.
External loop back mode

The FDCAN can be set in External loop back mode by programming TEST.LBCK to 1. In Loop Back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. Figure 776 shows the connection of transmit and receive signals FDCAN_TX and FDCAN_RX to the FDCAN in External loop back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data / remote frame) in Loop back mode. In this mode the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN_TX transmit pin.

Internal loop back mode

Internal loop back mode is entered by programming bits TEST.LBCK and CCCR.MON to 1. This mode can be used for a “Hot selftest”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins. In this mode, FDCAN_RX pin is disconnected from the FDCAN and FDCAN_TX pin is held recessive. Figure 776 shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of Internal loop back mode.

Figure 776. Pin control in Loop back mode

Timestamp generation

For timestamp generation the FDCAN supplies a 16-bit wrap-around counter. A prescaler TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1 ... 16). The counter is readable via TSCV[TCV]. A write access to register TSCV resets the counter to 0. When the timestamp counter wraps around interrupt flag IR[TSW] is set.

On start of frame reception/transmission the counter value is captured and stored into the timestamp section of a Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming bit TSCC.TSS, a 16-bit timestamp can be used.
**Debug mode behavior**

In debug mode the set / reset on read feature is automatically disabled during the debugger register access, and enabled during normal MCU operation.

**Timeout counter**

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as downcounter and uses the same prescaler controlled by TSCC[TCP] as the Timestamp Counter. The timeout counter is configured via register TOCC. The actual counter value can be read from TOCV[TOC]. The timeout counter can only be started while CCCR[INIT] = 0. It is stopped when CCCR[INIT] = 1, e.g. when the FDCAN enters Bus_Off state.

The operation mode is selected by TOCC[TOS]. When operating in Continuous mode, the counter starts when CCCR[INIT] is reset. A write to TOCV presets the counter to the value configured by TOCC[TOP] and continues downcounting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC[TOP]. Downcounting is started when the first FIFO element is stored. Writing to TOCV has no effect.

When the counter reaches 0, interrupt flag IR[TOO] is set. In Continuous mode, the counter is immediately restarted at TOCC[TOP].

*Note:* The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore the point in time where the timeout counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN core. If the baud rate switch feature in FDCAN is used, the timeout counter is clocked differently in arbitration and data fields.

**54.3.3 Message RAM**

The Message RAM has a 32-bit width, and the FDCAN module is configured to allocate up to 212 words in it. It is not necessary to configure each of the sections shown in Figure 777.

![Figure 777. Message RAM configuration](attachment:image.png)
When the FDCAN addresses the Message RAM, it addresses 32-bit words (aligned), not a single byte. The RAM addresses are 32-bit words, i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored.

In case of multiple instances the RAM start address for the FDCANn is computed by end address + 4 of FDCANn-1, and the FDCANn end address is computed by FDCANn start address + 0x0350 - 4.

As an example, for two instances:
- FDCAN1:
  - start address: 0x0000
  - end address: 0x034C (as in Figure 777)
- FDCAN2:
  - start address = 0x034C (FDCAN1 end address) + 4 = 0x0350
  - end address = 0x0350 (FDCAN2 start address) + 0x0350 - 4 = 0x069C.

**Rx handling**

The Rx handler controls the acceptance filtering, the transfer of received messages to Rx to one of the two Rx FIFOs, as well as the Rx FIFO Put and Get Indexes.

**Acceptance filter**

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and another for extended identifiers. These filters can be assigned to Rx FIFO 0 or Rx FIFO 1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element.

Following filter elements are not evaluated for this message.

The main features are:
- Each filter element can be configured as
  - range filter (from - to)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled/disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:
- Global Filter Configuration (RXGFC)
- Extended ID AND Mask (XIDAM)

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:
- Store received frame in FIFO 0 or FIFO 1
- Reject received frame
- Set High priority message interrupt flag IR[HPM]
- Set High priority message interrupt flag IR[HPM] and store received frame in FIFO 0 or FIFO 1.
Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx FIFO has been found, the Message Handler starts writing the received message data in 32-bit portions to the matching Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact:

- **Rx FIFO**
  Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see PSR.LEC and PSR.DLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Rx FIFO overwrite mode* have to be considered.

*Note:* When an accepted message is written to one of the two Rx FIFOs, the unmodified received identifier is stored independently from the used filter(s). The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

**Range filter**

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID and EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- **EFT = 00:** The Message ID of received frames is AND-ed with the Extended ID AND Mask (XIDAM) before the range filter is applied
- **EFT = 11:** The Extended ID AND Mask (XIDAM) is not used for range filtering

**Filter for dedicated IDs**

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID and EF1ID = EF2ID.

**Classic bit mask filter**

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A 0 bit at the filter mask masks out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are 0, all Message IDs match.

**Standard message ID filtering**

*Figure 778* shows the flow for standard message ID (11-bit Identifier) filtering. The standard message ID filter element is described in *Section 54.3.8*.

Controlled by the Global filter configuration (RXGFC) message ID, Remote transmission request bit (RTR), and the Identifier extension bit (IDE) of received frames are compared against the list of configured filter elements.
Figure 778. Standard Message ID filter path

- **Valid frame received**
- **Bit identifier**: 11-bit, 29-bit
- **Remote frame**: Yes, No
- **Receive filter list enabled**: Yes, No
- **RXGFC[RRFS]**: 0, 1
- **RXGFC[LSS[7:0]]**: > 0, = 0
- **Match filter element #0**: Yes, No
- **RXGFC[ANFS[1]]**: = 1, = 0
- **Target FIFO full**: Yes, No
- **Append to target FIFO**: Yes
- **Accept non-matching frames**: Yes, No
- **Acceptance or Rejection**: Reject, Accept
- **Discard frame**: Yes

- RXGFC[LSS[7:0]] > 0
- RXGFC[RRFS] = 0
- RXGFC[ANFS[1]] = 1
- RXGFC[ANFS[1]] = 0
Extended message ID filtering

Figure 779 shows the flow for extended message ID (29-bit identifier) filtering. The Extended Message ID filter element is described in Section 54.3.9.

Controlled by the Global filter configuration RXGFC and the Extended ID filter Configuration RXGFC message ID, Remote transmission request bit (RTR), and the Identifier extension bit (IDE) of received frames are compared against the list of configured filter elements.

Figure 779. Extended Message ID filter path

The Extended ID AND Mask (XIDAM) is AND-ed with the received identifier before the filter list is executed.
**Rx FIFOs**

Rx FIFO 0 and Rx FIFO 1 can hold up to three elements each.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see *Acceptance filter*. The Rx FIFO element is described in *Section 54.3.5*.

When an Rx FIFO full condition is signaled by IR[RFnF], no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded and interrupt flag IR[RFnL] is set.

When reading from an Rx FIFO, Rx FIFO Get Index RXFnS[FnGI] + FIFO Element Size has to be added to the corresponding Rx FIFO start address [FnSA].

**Rx FIFO blocking mode**

The Rx FIFO blocking mode is configured by RXGFC.FnOM = 0. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signaled by RXFnS.FnF = 1. In addition interrupt flag IR.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signaled by RXFnS.RFnL = 1. In addition interrupt flag IR.RFnL is set.

**Rx FIFO overwrite mode**

The Rx FIFO overwrite mode is configured by RXGFC.FnOM = 1.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signaled by RXFnS.FnF = 1, the next message accepted for the FIFO overwrites the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading of the Rx FIFO elements must start at least at get index + 1. This is because it can happen that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO.

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = 0).

**Tx handling**

The Tx Handler handles transmission requests for the Tx FIFO, and the Tx queue. It controls the transfer of transmit messages to the CAN core, the Put and Get Indices, and the Tx event FIFO. Up to three Tx buffers can be set up for message transmission. The CAN
message data field is configured to 64 bytes, Tx FIFO allocates eighteen 32-bit words for storage of a Tx element.

Table 578. Possible configurations for Frame transmission

<table>
<thead>
<tr>
<th>CCCR</th>
<th>Tx buffer element</th>
<th>Frame transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRSE</td>
<td>FDOE</td>
<td>FDF</td>
</tr>
<tr>
<td>Ignored</td>
<td>0</td>
<td>Ignored</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest Message ID) when the Tx buffer Request Pending register TXBRP is updated, or when a transmission has been started.

Transmit pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority must be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If, as an example, CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, it waits, after the first successful message transmission, for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by TXP bit in CCCR register. If the bit is set, the FDCAN, each time it has successfully transmitted a message, pauses for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is disabled (CCCR.TXP = 0).

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

Tx FIFO

Tx FIFO operation is configured by programming TXBC[TFQM] to 0. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index TXFQS[TFGI]. After each transmission the Get Index is incremented cyclically until the Tx
FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx buffers in the order these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO Free Level TXFQS[TFFL] as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (TXFQS[TFQF]= 1) is signaled. In this case no further messages must be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing 1 to the TXBAR bit related to the Tx buffer referenced by the Tx FIFO Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the Put Index. The transmissions are then requested via TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers must not exceed the number of free Tx buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx buffer referenced by the Get Index is canceled, the Get Index is incremented to the next Tx buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates eighteen 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx FIFO buffer is calculated by adding four times the Put Index TXFQS[TFQPI] (0 ... 2) to the Tx buffer Start Address TBSA.

**Tx queue**

Tx queue operation is configured by programming TXBC[TFQM] to 1. Messages stored in the Tx queue are transmitted starting with the message with the lowest Message ID (highest priority).

In case of mixing of standard and extended Message IDs, the standard Message IDs are compared to bits [28:18] of extended Message IDs.

In case that multiple queue buffers are configured with the same Message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx buffer referenced by the Put Index TXFQS[TFQPI]. An Add Request cyclically increments the Put Index to the next free Tx buffer. In case that the Tx queue is full (TXFQS[TFQF] = 1), the Put Index is not valid and no further message must be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

The application may use register TXBRP instead of the Put Index and may place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates eighteen 32-bit words in the Message RAM. Therefore the start address of the next available (free) Tx queue buffer is calculated by adding four times the Tx queue Put Index TXFQS[TFQPI] (0 ... 2) to the Tx buffer Start Address TBSA.
Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a Tx queue buffer the Host has to write a 1 to the corresponding bit position (= number of Tx buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register TXBCF to 1.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note: In case a pending transmission is canceled immediately before it has been started, there is a short time window where no transmission is started even if another message is pending in the node. This may enable another node to transmit a message that may have a priority lower than that of the second message in the node.

Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the Message Marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO is configured to three elements. The Tx event FIFO element is described in Tx FIFO.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling i.e. a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by IR[TEFF], no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and interrupt flag IR[TEFL] is set.

When reading from the Tx event FIFO, two times the Tx event FIFO Get Index TXEFS[EFGI] has to be added to the Tx event FIFO start address EFSA.

54.3.4 FIFO acknowledge handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index, see Section 54.4.23 and Section 54.4.25. Writing to the FIFO acknowledge index sets the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

1. When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.
2. When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO Get Index.
Due to the fact that the CPU has free access to the FDCAN Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High priority message from one of the two Rx FIFOs. In this case the FIFO Acknowledge Index must not be written because this would set the Get Index to a wrong position and also alters the FIFO Fill Level. In this case some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The FDCAN does not check for erroneous values.

54.3.5 FDCAN Rx FIFO element

Two Rx FIFOs are configured in the Message RAM. Each Rx FIFO section can be configured to store up to three received messages. The structure of an Rx FIFO element is described in Table 579, the description is provided in Table 580.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>24</th>
<th>16</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>ESI</td>
<td>XTD</td>
<td>RTR</td>
<td>ID[28:0]</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>DB3[7:0]</td>
<td>DB2[7:0]</td>
<td>DB1[7:0]</td>
<td>D[7:0]</td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td>DB7[7:0]</td>
<td>DB6[7:0]</td>
<td>DB5[7:0]</td>
<td>DB4[7:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Rn</td>
<td>DBm[7:0]</td>
<td>DBm-1[7:0]</td>
<td>DBm-2[7:0]</td>
<td>DBm-3[7:0]</td>
<td></td>
</tr>
</tbody>
</table>

The element size configured for storage of CAN FD messages is set to 64 bytes data field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0 Bit 31 ESI</td>
<td>Error state indicator</td>
</tr>
<tr>
<td>R0 Bit 30 XTD</td>
<td>Extended identifier</td>
</tr>
<tr>
<td>R0 Bit 29 RTR</td>
<td>Remote transmission request</td>
</tr>
<tr>
<td>R0 Bits 28:0 ID[28:0]</td>
<td>Identifier</td>
</tr>
</tbody>
</table>
### Table 580. Rx FIFO element description (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| R1 Bit 31 ANMF | Accepted non-matching frame  
Acceptedance of non-matching frames may be enabled via RXGFC[ANFS] and RXGFC[ANFE].  
– 0: Received frame matching filter index FIDX  
– 1: Received frame did not match any Rx filter element |
| R1 Bits 30:24 FIDX[6:0] | Filter index  
0-27=Index of matching Rx acceptance filter element (invalid if ANMF = 1).  
Range is 0 to RXGFC[LSS] - 1 or RXGFC[LSE] - 1. |
| R1 Bit 21 FDF | FD format  
– 0: Standard frame format  
– 1: FDCAN frame format (new DLC-coding and CRC) |
| R1 Bit 20 BRS | Bit rate switch  
– 0: Frame received without bit rate switching  
– 1: Frame received with bit rate switching |
| R1 Bits 19:16 DLC[3:0] | Data length code  
– 0-8: Classic CAN + CAN FD: received frame has 0-8 Data bytes  
– 9-15: Classic CAN: received frame has 8 Data bytes  
– 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 Data bytes |
| R1 Bits 15:0 RXTS[15:0] | Rx timestamp  
Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP]. |
| R2 Bits 31:24 DB3[7:0] | Data byte 3 |
| R2 Bits 23:16 DB2[7:0] | Data byte 2 |
| R2 Bits 15:8 DB1[7:0] | Data byte 1 |
| R2 Bits 7:0 D[7:0] | Data byte 0 |
| R3 Bits 31:24 DB7[7:0] | Data byte 7 |
| R3 Bits 23:16 DB6[7:0] | Data byte 6 |
| R3 Bits 15:8 DB5[7:0] | Data byte 5 |
| R3 Bits 7:0 DB4[7:0] | Data byte 4 |
| ... | ... |
| Rn Bits 31:24 DBm[7:0] | Data byte m |
54.3.6 FDCAN Tx buffer element

The Tx buffers section (three elements) can be configured to hold Tx FIFO or Tx queue. The Tx Handler distinguishes between Tx FIFO and Tx queue using the Tx buffer configuration FDCAN_TXBC.TFQM. The element size is configured for storage of CAN FD messages with up to 64 bytes data.

<table>
<thead>
<tr>
<th>Table 580. Rx FIFO element description (continued)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
</tr>
<tr>
<td>Rn Bits 23:16 DBm-1[7:0]</td>
</tr>
<tr>
<td>Rn Bits 15:8 DBm-2[7:0]</td>
</tr>
<tr>
<td>Rn Bits 7:0 DBm-3[7:0]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 581. Tx buffer and FIFO element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 31 24 23 16 15 8 7 0</td>
</tr>
<tr>
<td>T0 ESI XTD RTR</td>
</tr>
<tr>
<td>T2 DB3[7:0] DB2[7:0] DB1[7:0] D[7:0]</td>
</tr>
<tr>
<td>T3 DB7[7:0] DB6[7:0] DB5[7:0] DB4[7:0]</td>
</tr>
<tr>
<td>Tn DBm[7:0] DBm-1[7:0] DBm-2[7:0] DBm-3[7:0]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 582. Tx buffer element description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
</tr>
<tr>
<td>T0 Bit 31 ESI(1)</td>
</tr>
<tr>
<td>– 0: ESI bit in CAN FD format depends only on error passive flag</td>
</tr>
<tr>
<td>– 1: ESI bit in CAN FD format transmitted recessive</td>
</tr>
<tr>
<td>T0 Bit 30 XTD</td>
</tr>
<tr>
<td>– 0: 11-bit standard identifier</td>
</tr>
<tr>
<td>– 1: 29-bit extended identifier</td>
</tr>
<tr>
<td>T0 Bit 29 RTR(2)</td>
</tr>
<tr>
<td>– 0: Transmit data frame</td>
</tr>
<tr>
<td>– 1: Transmit remote frame</td>
</tr>
<tr>
<td>T0 Bits 28:0 ID[28:0]</td>
</tr>
<tr>
<td>Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].</td>
</tr>
<tr>
<td>T1 Bits 31:24 MM[7:0]</td>
</tr>
<tr>
<td>Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.</td>
</tr>
</tbody>
</table>
### Table 582. Tx buffer element description (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1 Bit 23</td>
<td>Event FIFO control</td>
</tr>
<tr>
<td>EFC</td>
<td>– 0: Do not store Tx events</td>
</tr>
<tr>
<td></td>
<td>– 1: Store Tx events</td>
</tr>
<tr>
<td>T1 Bit 21</td>
<td>FD format</td>
</tr>
<tr>
<td>FDF</td>
<td>– 0: Frame transmitted in Classic CAN format</td>
</tr>
<tr>
<td></td>
<td>– 1: Frame transmitted in CAN FD format</td>
</tr>
<tr>
<td>T1 Bit 20</td>
<td>Bit rate switching</td>
</tr>
<tr>
<td>BRS(3)</td>
<td>– 0: CAN FD frames transmitted without bit rate switching</td>
</tr>
<tr>
<td></td>
<td>– 1: CAN FD frames transmitted with bit rate switching</td>
</tr>
<tr>
<td>T1 Bits 19:16</td>
<td>Data length code</td>
</tr>
<tr>
<td>DLC[3:0]</td>
<td>– 0 - 8: Classic CAN + CAN FD: received frame has 0-8 Data bytes</td>
</tr>
<tr>
<td></td>
<td>– 9 - 15: Classic CAN: received frame has 8 Data bytes</td>
</tr>
<tr>
<td></td>
<td>– 9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 Data bytes</td>
</tr>
<tr>
<td>T2 Bits 31:24</td>
<td>Data byte 3</td>
</tr>
<tr>
<td>DB3[7:0]</td>
<td></td>
</tr>
<tr>
<td>T2 Bits 23:16</td>
<td>Data byte 2</td>
</tr>
<tr>
<td>DB2[7:0]</td>
<td></td>
</tr>
<tr>
<td>T2 Bits 15:8</td>
<td>Data byte 1</td>
</tr>
<tr>
<td>DB1[7:0]</td>
<td></td>
</tr>
<tr>
<td>T2 Bits 7:0</td>
<td>Data byte 0</td>
</tr>
<tr>
<td>D[7:0]</td>
<td></td>
</tr>
<tr>
<td>T3 Bits 31:24</td>
<td>Data byte 7</td>
</tr>
<tr>
<td>DB7[7:0]</td>
<td></td>
</tr>
<tr>
<td>T3 Bits 23:16</td>
<td>Data byte 6</td>
</tr>
<tr>
<td>DB6[7:0]</td>
<td></td>
</tr>
<tr>
<td>T3 Bits 15:8</td>
<td>Data byte 5</td>
</tr>
<tr>
<td>DB5[7:0]</td>
<td></td>
</tr>
<tr>
<td>T3 Bits 7:0</td>
<td>Data byte 4</td>
</tr>
<tr>
<td>DB4[7:0]</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Tn Bits 31:24</td>
<td>Data byte m</td>
</tr>
<tr>
<td>DBm[7:0]</td>
<td></td>
</tr>
<tr>
<td>Tn Bits 23:16</td>
<td>Data byte m-1</td>
</tr>
<tr>
<td>DBm-1[7:0]</td>
<td></td>
</tr>
<tr>
<td>Tn Bits 15:8</td>
<td>Data byte m-2</td>
</tr>
<tr>
<td>DBm-2[7:0]</td>
<td></td>
</tr>
<tr>
<td>Tn Bits 7:0</td>
<td>Data byte m-3</td>
</tr>
<tr>
<td>DBm-3[7:0]</td>
<td></td>
</tr>
</tbody>
</table>

1. The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node always transmits the ESI bit recessive.
2. When RTR = 1, the FDCAN transmits a remote frame according to ISO11898-1, even if CCCR.FDOE enables the transmission in CAN FD format.

3. Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled CCCR.FDOE = 1. Bit BRS is only evaluated when in addition CCCR.BRSE = 1.

### 54.3.7 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx event FIFO can be obtained from register TXEFS.

**Table 583. Tx event FIFO element**

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>ESI</td>
<td>XTD</td>
<td>RTR</td>
<td>ID[28:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 584. Tx event FIFO element description**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| E0 Bit 31 ESI | Error state indicator  
– 0: Transmitting node is error active  
– 1: Transmitting node is error passive |
| E0 Bit 30 XTD | Extended identifier  
– 0: 11-bit standard identifier  
– 1: 29-bit extended identifier |
| E0 Bit 29 RTR | Remote transmission request  
– 0: Transmit data frame  
– 1: Transmit remote frame |
| E0 Bits 28:0 ID[28:0] | Identifier  
Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18]. |
| E1 Bits 31:24 MM[7:0] | Message marker  
Copied from Tx buffer into Tx event FIFO element for identification of Tx message status. |
| E1 Bits 23:22 EFC | Event type  
– 00: Reserved  
– 01: Tx event  
– 10: Transmission in spite of cancellation (always set for transmissions in DAR mode)  
– 11: Reserved |
| E1 Bit 21 EDL | Extended data length  
– 0: Standard frame format  
– 1: FDCAN frame format (new DLC-coding and CRC) |
| E1 Bit 20 BRS | Bit rate switching  
– 0: Frame transmitted without bit rate switching  
– 1: Frame transmitted with bit rate switching |
### 54.3.8 FDCAN Standard message ID filter element

Up to 28 filter elements can be configured for 11-bit standard IDs. When accessing a Standard message ID filter element, its address is the Filter list standard start address FLSSA plus the index of the filter element (0 … 27).

#### Table 585. Standard message ID filter element

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
</table>

#### Table 586. Standard message ID filter element field description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Bit 31:30 SFT[1:0](1) | Standard filter type  
– 00: Range filter from SFID1 to SFID2  
– 01: Dual ID filter for SFID1 or SFID2  
– 10: Classic filter: SFID1 = filter, SFID2 = mask  
– 11: Filter element disabled |
| Bit 29:27 SFEC[2:0] | Standard filter element configuration  
All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = 100, 101 or 110 a match sets interrupt flag IR.HPM and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match.  
– 000: Disable filter element  
– 001: Store in Rx FIFO 0 if filter matches  
– 010: Store in Rx FIFO 1 if filter matches  
– 011: Reject ID if filter matches  
– 100: Set priority if filter matches  
– 101: Set priority and store in FIFO 0 if filter matches  
– 110: Set priority and store in FIFO 1 if filter matches  
– 111: Not used |
| Bits 26:16 SFID1[10:0] | Standard filter ID 1  
First ID of standard ID filter element. |
| Bits 10:0 SFID2[10:0] | Standard filter ID 2  
Second ID of standard ID filter element. |

---

Table 584. Tx event FIFO element description (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| T1 Bits 19:16 DLC[3:0] | Data length code  
0 - 8: Frame with 0-8 Data bytes transmitted  
9 - 15: Frame with 8 Data bytes transmitted |
| E1 Bits 15:0 TXTS[15:0] | Tx Timestamp  
Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC[TCP]. |
1. With SFT = 11 the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC = 000).

Note: In case a reserved value is configured, the filter element is considered disabled.

**54.3.9 FDCAN Extended message ID filter element**

Up to 8 filters element can be configured for 29-bit extended IDs. When accessing an Extended message ID filter element, its address is the Filter list extended start address FLESA plus two times the index of the filter element (0 … 7).

**Table 587. Extended message ID filter element**

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>F0</td>
<td>EFEC[2:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>EFTI[1:0]</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EFID2[28:0]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 588. Extended message ID filter element field description**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| F0 Bits 31:29 EFEC[2:0] | Extended filter element configuration  
All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = 100, 101 or 110 a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match.  
– 000: Disable filter element  
– 001: Store in Rx FIFO 0 if filter matches  
– 010: Store in Rx FIFO 1 if filter matches  
– 011: Reject ID if filter matches  
– 100: Set priority if filter matches  
– 101: Set priority and store in FIFO 0 if filter matches  
– 110: Set priority and store in FIFO 1 if filter matches  
– 111: Not used |
| F0 Bits 28:0 EFID1[28:0] | Extended filter ID 1  
First ID of extended ID filter element. When filtering for Rx FIFO, this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism. |
| F1 Bits 31:30 EFTI[1:0] | Extended filter type  
– 00: Range filter from EF1ID to EF2ID (EF2ID >= EF1ID)  
– 01: Dual ID filter for EF1ID or EF2ID  
– 10: Classic filter: EF1ID = filter, EF2ID = mask  
– 11: Range filter from EF1ID to EF2ID (EF2ID >= EF1ID), XIDAM mask not applied |
| F1 Bit 29 | Not used |
| F1 Bits 28:0 EFID2[28:0] | Extended filter ID 2  
Second ID of extended ID filter element. |
# FDCAN registers

## 54.4.1 FDCAN core release register (FDCAN_CREL)

Address offset: 0x0000  
Reset value: 0x3214 1218

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
</tbody>
</table>

**Bits 31:28**  REL[3:0]: 3  
**Bits 27:24**  STEP[3:0]: 2  
**Bits 23:20**  SUBSTEP[3:0]: 1  
**Bits 19:16**  YEAR[3:0]: 4

<table>
<thead>
<tr>
<th>MON[7:0]</th>
<th>DAY[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**Bits 31:28**  MON[7:0]: 12  
**Bits 7:0**  DAY[7:0]: 18

## 54.4.2 FDCAN endian register (FDCAN_ENDN)

Address offset: 0x0004  
Reset value: 0x8765 4321

<table>
<thead>
<tr>
<th>ETV[31:16]</th>
<th>ETV[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
</tr>
</tbody>
</table>

**Bits 31:0**  ETV[31:0]: Endianness test value  
The endianness test value is 0x8765 4321.

**Note:** The register read must give the reset value to ensure no endianness issue.
54.4.3  FDCAN data bit timing and prescaler register (FDCAN_DBTP)

Address offset: 0x000C
Reset value: 0x0000 0A33

This register is only writable if bits CCCR.CCE and CCCR.INIT are set. The CAN time quantum may be programmed in the range of 1 to 32 FDCAN clock periods. tq = (DBRP + 1) FDCAN clock period.

DTSEG1 is the sum of Prop_Seg and Phase_Seg1. DTSEG2 is Phase_Seg2. Therefore the length of the bit time is (programmed values) [DTSEG1 + DTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The Information Processing Time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

| Bit 31:24 | Reserved, must be kept at reset value. |
| Bit 23 | **TDC**: Transceiver delay compensation |
| 0: Transceiver delay compensation disabled |
| 1: Transceiver delay compensation enabled |
| Bit 20:16 | **DBRP[4:0]**: Data bit rate prescaler |
| The value by which the oscillator frequency is divided to generate the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 31. The hardware interpreters this value as the value programmed plus 1. |
| Bit 15:13 | Reserved, must be kept at reset value. |
| Bit 12:8 | **DTSEG1[4:0]**: Data time segment before sample point |
| Valid values are 0 to 31. The value used by the hardware is the one programmed, incremented by 1, i.e. tBS1 = (DTSEG1 + 1) x tq. |
| Bit 7:4 | **DTSEG2[3:0]**: Data time segment after sample point |
| Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, i.e. tBS2 = (DTSEG2 + 1) x tq. |
| Bit 3:0 | **DSJW[3:0]**: Synchronization jump width |
| Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1: tSJW = (DSJW + 1) x tq. |

**Note:**  With an FDCAN clock of 8 MHz, the reset value 0x00000A33 configures the FDCAN for a fast bitrate of 500 kbit/s.

**Note:**  The data phase bit rate must be higher to or equal to the nominal bit rate.
54.4.4 FDCAN test register (FDCAN_TEST)

Write access to the this register is enabled by setting bit CCCR[TEST] to 1. All register functions are set to their reset values when bit CCCR[TEST] is reset.

Loop Back mode and software control of Tx pin FDCANx_TX are hardware test modes. Programming TX differently from 00 may disturb the message transfer on the CAN bus.

Address offset: 0x0010
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>RX</td>
<td>TX[1:0]</td>
<td>LBCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bit 7  RX: Receive pin
  Monitors the actual value of pin FDCANx_RX
  0: The CAN bus is dominant (FDCANx_RX = 0)
  1: The CAN bus is recessive (FDCANx_RX = 1)

Bits 6:5  TX[1:0]: Control of transmit pin
  00: Reset value, FDCANx_TX TX is controlled by the CAN core, updated at the end of the CAN bit time
  01: Sample point can be monitored at pin FDCANx_TX
  10: Dominant (0) level at pin FDCANx_TX
  11: Recessive (1) at pin FDCANx_TX

Bit 4  LBCK: Loop back mode
  0: Reset value, Loop Back mode is disabled
  1: Loop Back mode is enabled (see Power down (Sleep mode))

Bits 3:0  Reserved, must be kept at reset value.
**54.4.5 FDCAN RAM watchdog register (FDCAN_RWD)**

The RAM Watchdog monitors the READY output of the Message RAM. A Message RAM access starts the Message RAM Watchdog Counter with the value configured by the RWD[WDC] bits.

The counter is reloaded with RWD[WDC] bits when the Message RAM signals successful completion by activating its READY output. In case there is no response from the Message RAM until the counter has counted down to 0, the counter stops and interrupt flag IR[WDI] bit is set. The RAM Watchdog Counter is clocked by the fdcan_pclk clock.

Address offset: 0x0014

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WDV[7:0]</th>
<th>WDC[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **WDV[7:0]**: Watchdog value

Actual message RAM watchdog counter value.

Bits 7:0 **WDC[7:0]**: Watchdog configuration

Start value of the message RAM watchdog counter. With the reset value of 00, the counter is disabled.

These are protected write (P) bits, write access is possible only when the bit 1 [CCE] and bit 0 [INIT] of FDCAN_CCCR register are set to 1.

**54.4.6 FDCAN CC control register (FDCAN_CCCR)**

Address offset: 0x0018

Reset value: 0x0000 0001

For details about setting and resetting of single bits, see *Software initialization*.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NISO</th>
<th>TXP</th>
<th>EFBI</th>
<th>PXHD</th>
<th>BRSE</th>
<th>FDOE</th>
<th>TEST</th>
<th>DAR</th>
<th>MON</th>
<th>CSR</th>
<th>CSA</th>
<th>ASM</th>
<th>CCE</th>
<th>INIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>r</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NISO**: Non ISO operation

If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.

0: CAN FD frame format according to ISO11898-1

1: CAN FD frame format according to Bosch CAN FD Specification V1.0
Bit 14 **TXP:**
If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.
0: disabled
1: enabled

Bit 13 **EFBI:** Edge filtering during bus integration
0: Edge filtering disabled
1: Two consecutive dominant tq required to detect an edge for hard synchronization

Bit 12 **PXHD:** Protocol exception handling disable
0: Protocol exception handling enabled
1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BRSE:** FDCAN bit rate switching
0: Bit rate switching for transmissions disabled
1: Bit rate switching for transmissions enabled

Bit 8 **FDOE:** FD operation enable
0: FD operation disabled
1: FD operation enabled

Bit 7 **TEST:** Test mode enable
0: Normal operation, register TEST holds reset values
1: Test Mode, write access to register TEST enabled

Bit 6 **DAR:** Disable automatic retransmission
0: Automatic retransmission of messages not transmitted successfully enabled
1: Automatic retransmission disabled

Bit 5 **MON:** Bus monitoring mode
Bit MON can only be set by software when both CCE and INIT are set to 1. The bit can be reset by the Host at any time.
0: Bus monitoring mode disabled
1: Bus monitoring mode enabled

Bit 4 **CSR:** Clock stop request
0: No clock stop requested
1: Clock stop requested. When clock stop is requested, first INIT and then CSA is set after all pending transfer requests have been completed and the CAN bus reached idle.

Bit 3 **CSA:** Clock stop acknowledge
0: No clock stop acknowledged
1: FDCAN may be set in power down by stopping APB clock and kernel clock.
Bit 2 **ASM**: ASM restricted operation mode

The restricted operation mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the Restricted operation Mode after it has received a valid frame. In the optional Restricted operation Mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by software when both CCE and INIT are set to 1. The bit can be reset by the software at any time.

0: Normal CAN operation
1: Restricted operation Mode active

Bit 1 **CCE**: Configuration change enable

0: The CPU has no write access to the protected configuration registers.
1: The CPU has write access to the protected configuration registers (while CCCR.INIT = 1).

Bit 0 **INIT**: Initialization

0: Normal operation
1: Initialization started

**Note**: Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

### 54.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)

Address offset: 0x001C

Reset value: 0x0600 0A03

This register is only writable if bits CCCR[CCE] and CCCR[INIT] are set. The CAN bit time may be programed in the range of 4 to 81 tq. The CAN time quantum may be programmed in the range of \([1 \ldots 1024]\) FDCAN kernel clock periods.

\[tq = (BRP + 1) \text{ FDCAN clock period fdcan clk}\]

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2. Therefore the length of the bit time is (programmed values) \([\text{NTSEG1} + \text{NTSEG2} + 3]\) tq or (functional values) \([\text{Sync_Seg} + \text{Prop_Seg} + \text{Phase_Seg1} + \text{Phase_Seg2}]\) tq.

The Information Processing Time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.
Note: With a CAN kernel clock of 48 MHz, the reset value of 0x06000A03 configures the FDCAN for a bit rate of 3 Mbit/s.

54.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC)

Address offset: 0x0020
Reset value: 0x0000 0000

| Bit 31:20 | Reserved, must be kept at reset value. |
| Bit 19:16 | TCP[3:0]: Timestamp counter prescaler |
| Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 … 16]. |
| The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. |
| In CAN FD mode the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS = 10). |
| These are protected write (P) bits, write access is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1. |

| Bit 15:2 | Reserved, must be kept at reset value. |
| Bit 7 | Reserved, must be kept at reset value. |
| Bit 6:0 | TSS[1:0]: |
| rw rw |

Bits 31:25 NSJW[6:0]: Nominal (re)synchronization jump width
Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that the used value is the one programmed incremented by one.
These are protected write (P) bits, write access is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bits 24:16 NBRP[8:0]: Bit rate prescaler
Value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
These are protected write (P) bits, write access is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bits 15:8 NTSEG1[7:0]: Nominal time segment before sample point
Valid values are 0 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
These are protected write (P) bits, write access is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 NTSEG2[6:0]: Nominal time segment after sample point
Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
54.4.9 **FDCAN timestamp counter value register (FDCAN_TSCV)**

Address offset: 0x0024

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
</table>

TSC[15:0] = 0x00

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSC[15:0]: Timestamp counter**

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). When TSCC[TSS] = 01, the timestamp counter is incremented in multiples of CAN bit times [1 … 16] depending on the configuration of TSCC[TCP]. A wrap around sets interrupt flag IR[TSW]. Write access resets the counter to 0.

When TSCC.TSS = 10, TSC reflects the external timestamp counter value. A write access has no impact.

**Note:** A “wrap around” is a change of the Timestamp Counter value from non-0 to 0 that is not caused by write access to TSCV.

54.4.10 **FDCAN timeout counter configuration register (FDCAN_TOCC)**

Address offset: 0x0028

Reset value: 0xFFFF 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

TOP[15:0] = 0x00

Bits 31:16 **TOP[15:0]: Timeout period**

Start value of the timeout counter (down-counter). Configures the timeout period.

Bits 15:3 Reserved, must be kept at reset value.
54.4.11 **FDCAN timeout counter value register (FDCAN_TOCV)**

Address offset: 0x002C

Reset value: 0x0000 FFFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w</td>
<td>rc_w</td>
<td>rc_w</td>
<td>rc_w</td>
<td>rc_w</td>
<td>rc_w</td>
<td>rc_w</td>
<td>rc_w</td>
<td>rc_w</td>
<td>rc_w</td>
<td>rc_w</td>
<td>rc_w</td>
<td>rc_w</td>
<td>rc_w</td>
<td>rc_w</td>
<td>rc_w</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TOC[15:0]:** Timeout counter

The timeout counter is decremented in multiples of CAN bit times [1 … 16] depending on the configuration of TSCC.TCP. When decremented to 0, interrupt flag IR.TO0 is set and the timeout counter is stopped. Start and reset/restart conditions are configured via TOCC.TOS.

54.4.12 **FDCAN error counter register (FDCAN_ECR)**

Address offset: 0x0040

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>ce</td>
<td>c_m</td>
</tr>
</tbody>
</table>

Bits 31:23 Reserved

Bits 22:20 **REC[8:0]:** Error counter

Bits 19:17 **TEC[7:0]:** Time error counter
Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CEL[7:0]: CAN error logging**

The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag IR[EL0].

Access type is RX: reset on read.

Bit 15 **RP**: Receive error passive

0: The receive error counter is below the error passive level of 128.
1: The receive error counter has reached the error passive level of 128.

Bits 14:8 **REC[6:0]: Receive error counter**

Actual state of the receive error counter, values between 0 and 127.

Bits 7:0 **TEC[7:0]: Transmit error counter**

Actual state of the transmit error counter, values between 0 and 255.

When CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

### 54.4.13 FDCAN protocol status register (FDCAN_PSR)

Address offset: 0x0044

Reset value: 0x0000 0707

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TDCV[6:0]: Transmitter delay compensation value**

Position of the secondary sample point, defined by the sum of the measured delay from FDCAN_TX to FDCAN_RX and TDCR.TDCO. The SSP position is, in the data phase, the number of minimum time quanta (mtq) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PXE**: Protocol exception event

0: No protocol exception event occurred since last read access
1: Protocol exception event occurred

Bit 13 **REDL**: Received FDCAN message

This bit is set independent of acceptance filtering.
0: Since this bit was reset by the CPU, no FDCAN message has been received.
1: Message in FDCAN format with EDL flag set has been received.

Access type is RX: reset on read.
Bit 12 **RBRS**: BRS flag of last received FDCAN message
   This bit is set together with REDL, independent of acceptance filtering.
   0: Last received FDCAN message did not have its BRS flag set.
   1: Last received FDCAN message had its BRS flag set.
   Access type is RX: reset on read.

Bit 11 **RESI**: ESI flag of last received FDCAN message
   This bit is set together with REDL, independent of acceptance filtering.
   0: Last received FDCAN message did not have its ESI flag set.
   1: Last received FDCAN message had its ESI flag set.
   Access type is RX: reset on read.

Bits 10:8 **DLEC[2:0]**: Data last error code
   Type of last error that occurred in the data phase of a FDCAN format frame with its BRS flag set. Coding is the same as for LEC. This field is cleared to 0 when a FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.
   Access type is RS: set on read.

Bit 7 **BO**: Bus_Off status
   0: The FDCAN is not Bus_Off.
   1: The FDCAN is in Bus_Off state.

Bit 6 **EW**: Warning Status
   0: Both error counters are below the Error_Warning limit of 96.
   1: At least one of error counter has reached the Error_Warning limit of 96.

Bit 5 **EP**: Error passive
   0: The FDCAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected.
   1: The FDCAN is in the Error_Passive state.

Bits 4:3 **ACT[1:0]**: Activity
   Monitors the module’s CAN communication state.
   00: Synchronizing: node is synchronizing on CAN communication.
   01: Idle: node is neither receiver nor transmitter.
   10: Receiver: node is operating as receiver.
   11: Transmitter: node is operating as transmitter.
Bits 2:0 **LEC[2:0]**: Last error code

The LEC indicates the type of the last error to occur on the CAN bus. This field is cleared to 0 when a message has been transferred (reception or transmission) without error.

- **000**: No Error: No error occurred since LEC has been reset by successful reception or transmission.
- **001**: Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
- **010**: Form Error: A fixed format part of a received frame has the wrong format.
- **011**: AckError: The message transmitted by the FDCAN was not acknowledged by another node.
- **100**: Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.
- **101**: Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
- **110**: CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.
- **111**: NoChange: Any read access to the Protocol status register re-initializes the LEC to ‘7’. When the LEC shows the value ‘7’, no CAN bus event was detected since the last CPU read access to the Protocol status register.

Access type is RS: set on read.

Note: When a frame in FDCAN format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) is shown in FLEC instead of LEC. An error in a fixed stuff bit of a FDCAN CRC sequence is shown as a Form Error, not Stuff Error.

Note: The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting CCCR[INIT]. If the device goes Bus_Off, it sets CCCR.INIT of its own, stopping all bus activities. Once CCCR[INIT] has been cleared by the CPU, the device then waits for 129 occurrences of Bus_Idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters are reset. During the waiting time after the reset of CCCR[INIT], each time a sequence of 11 recessive bits has been monitored, a Bit0 Error code is written to PSR[LEC], enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR[REC] is used to count these sequences.

### 54.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR)

Address offset: 0x0048

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res</td>
<td>TDC[6:0]</td>
<td>Res</td>
<td>TDCF[5:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position.

Writing a 0 has no effect. A hard reset clears the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050
Reset value: 0x0000 0000

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 ARA: Access to reserved address
0: No access to reserved address occurred
1: Access to reserved address occurred

Bit 22 PED: Protocol error in data phase (data bit time is used)
0: No protocol error in data phase
1: Protocol error in data phase detected (PSR.DLEC different from 0,7)

Bit 21 PEA: Protocol error in arbitration phase (nominal bit time is used)
0: No protocol error in arbitration phase
1: Protocol error in arbitration phase detected (PSR.LEC different from 0,7)

Bit 20 WDI: Watchdog interrupt
0: No message RAM watchdog event occurred
1: Message RAM watchdog event due to missing READY
Bit 19  **BO**: Bus_Off status  
0: Bus_Off status unchanged  
1: Bus_Off status changed  

Bit 18  **EW**: Warning status  
0: Error_Warning status unchanged  
1: Error_Warning status changed  

Bit 17  **EP**: Error passive  
0: Error_Passive status unchanged  
1: Error_Passive status changed  

Bit 16  **ELO**: Error logging overflow  
0: CAN error logging counter did not overflow.  
1: Overflow of CAN error logging counter occurred.  

Bit 15  **TOO**: Timeout occurred  
0: No timeout  
1: Timeout reached  

Bit 14  **MRAF**: Message RAM access failure  
The flag is set when the Rx handler:  
1. has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.  
2. was unable to write a message to the message RAM. In this case message storage is aborted.  
In both cases the FIFO put index is not updated. The partly stored message is overwritten when the next message is stored to this location.  
The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the FDCAN is switched into Restricted operation Mode (see Restricted operation mode). To leave Restricted operation Mode, the Host CPU has to reset CCCR.ASM.  
0: No Message RAM access failure occurred  
1: Message RAM access failure occurred  

Bit 13  **TSW**: Timestamp wraparound  
0: No timestamp counter wrap-around  
1: Timestamp counter wrapped around  

Bit 12  **TEFL**: Tx event FIFO element lost  
0: No Tx event FIFO element lost  
1: Tx event FIFO element lost  

Bit 11  **TEFF**: Tx event FIFO full  
0: Tx event FIFO Not full  
1: Tx event FIFO full  

Bit 10  **TEFN**: Tx event FIFO New Entry  
0: Tx event FIFO unchanged  
1: Tx handler wrote Tx event FIFO element.  

Bit 9  **TFE**: Tx FIFO empty  
0: Tx FIFO non-empty  
1: Tx FIFO empty
Bit 8  **TCF**: Transmission cancellation finished
   0: No transmission cancellation finished
   1: Transmission cancellation finished

Bit 7  **TC**: Transmission completed
   0: No transmission completed
   1: Transmission completed

Bit 6  **HPM**: High-priority message
   0: No high-priority message received
   1: High-priority message received

Bit 5  **RF1L**: Rx FIFO 1 message lost
   0: No Rx FIFO 1 message lost
   1: Rx FIFO 1 message lost

Bit 4  **RF1F**: Rx FIFO 1 full
   0: Rx FIFO 1 not full
   1: Rx FIFO 1 full

Bit 3  **RF1N**: Rx FIFO 1 new message
   0: No new message written to Rx FIFO 1
   1: New message written to Rx FIFO 1

Bit 2  **RF0L**: Rx FIFO 0 message lost
   0: No Rx FIFO 0 message lost
   1: Rx FIFO 0 message lost

Bit 1  **RF0F**: Rx FIFO 0 full
   0: Rx FIFO 0 not full
   1: Rx FIFO 0 full

Bit 0  **RF0N**: Rx FIFO 0 new message
   0: No new message written to Rx FIFO 0
   1: New message written to Rx FIFO 0

54.4.16  **FDCAN interrupt enable register (FDCAN_IE)**

The settings in the interrupt enable register determine which status changes in the interrupt
register are signaled on an interrupt line.

Address offset: 0x00054

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>TOOE</td>
<td>MRAFE</td>
<td>TSWE</td>
<td>TEFLE</td>
<td>TEFFE</td>
<td>TEFNE</td>
<td>TFEF</td>
<td>TCFE</td>
<td>TCE</td>
<td>HPMIE</td>
<td>RF1LE</td>
<td>RF1FE</td>
<td>RF1NE</td>
<td>RF0LE</td>
<td>RF0FE</td>
<td>RF0NE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:24  Reserved, must be kept at reset value.

   Bit 23  **ARAE**: Access to reserved address enable
   Bit 22  **PEDE**: Protocol error in data phase enable
<table>
<thead>
<tr>
<th>Bit 21</th>
<th>PEAE: Protocol error in arbitration phase enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 20</td>
<td>WDIE: Watchdog interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td>Bit 19</td>
<td>BOE: Bus_Off status</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td>Bit 18</td>
<td>EWE: Warning status interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td>Bit 17</td>
<td>EPE: Error passive interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td>Bit 16</td>
<td>ELOE: Error logging overflow interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td>Bit 15</td>
<td>TOOE: Timeout occurred interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td>Bit 14</td>
<td>MRAFE: Message RAM access failure interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td>Bit 13</td>
<td>TSWE: Timestamp wraparound interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td>Bit 12</td>
<td>TEFLE: Tx event FIFO element lost interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td>Bit 11</td>
<td>TEFFE: Tx event FIFO full interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td>Bit 10</td>
<td>TEFNE: Tx event FIFO new entry interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td>Bit 9</td>
<td>TFEE: Tx FIFO empty interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td>Bit 8</td>
<td>TCFE: Transmission cancellation finished interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td>Bit 7</td>
<td>TCE: Transmission completed interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
</tbody>
</table>
54.4.17 **FDCAN interrupt line select register (FDCAN ILS)**

This register assigns an interrupt generated by a specific group of interrupt flag from the Interrupt register to one of the two module interrupt lines. For interrupt generation the respective interrupt line has to be enabled via ILE[EINT0] and ILE[EINT1].

Address offset: 0x0058

Reset value: 0x0000 0000

| Bit 0: RF0NE | Rx FIFO 0 new message interrupt enable |
| Bit 1: RF0FE | Rx FIFO 0 full interrupt enable |
| Bit 2: RF1FE | Rx FIFO 1 full interrupt enable |
| Bit 3: RF1NE | Rx FIFO 1 new message interrupt enable |
| Bit 5: RF1LE | Rx FIFO 1 message lost interrupt enable |
| Bit 6: HPME | High-priority message interrupt enable |

Bits 31:7 Reserved, must be kept at reset value.

**Bit 6 PERR:** Protocol error grouping the following interruption
- ARAL: Access to reserved address line
- PEDL: Protocol error in data phase line
- PEAL: Protocol error in arbitration phase line
- WDIL: Watchdog interrupt line
- BOL: Bus_Off status
- EWL: Warning status interrupt line
54.4.18 FDCAN interrupt line enable register (FDCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming bits EINT0 and EINT1.

Address offset: 0x005C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:2: Reserved, must be kept at reset value.

Bit 1 **EINT1**: Enable interrupt line 1
- 0: Interrupt line fdcan_intr0_it disabled
- 1: Interrupt line fdcan_intr0_it enabled

Bit 0 **EINT0**: Enable interrupt line 0
- 0: Interrupt line fdcan_intr1_it disabled
- 1: Interrupt line fdcan_intr1_it enabled
54.4.19 FDCAN global filter configuration register (FDCAN_RXGFC)

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in Figure 778 and Figure 779.

Address offset: 0x0080
Reset value: 0x0000 0000

|   |   |   |   |   | LSE[3:0] |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 31| 30| 29| 28| 27| 26| 25| 24| 23| 22| 21| 20| 19| 18| 17| 16| 15| 14| 13| 12| 11| 10|  9|  8|  7|  6|  5|  4|  3|  2|  1|  0|
|   |   |   |   |   | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **LSE[3:0]:** List size extended
- 0: No extended message ID filter
- 1 to 8: Number of extended message ID filter elements
- >8: Values greater than 8 are interpreted as 8.
  These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **LSS[4:0]:** List size standard
- 0: No standard message ID filter
- 1 to 28: Number of standard message ID filter elements
- >28: Values greater than 28 are interpreted as 28.
  These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **F0OM:** FIFO 0 operation mode (overwrite or blocking)
- This is a protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bit 8 **F1OM:** FIFO 1 operation mode (overwrite or blocking)
- This is a protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ANFS[1:0]:** Accept Non-matching frames standard
- Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.
  - 00: Accept in Rx FIFO 0
  - 01: Accept in Rx FIFO 1
  - 10: Reject
  - 11: Reject
  These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.
Bits 3:2 **ANFE[1:0]**: Accept non-matching frames extended

Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.

- 00: Accept in Rx FIFO 0
- 01: Accept in Rx FIFO 1
- 10: Reject
- 11: Reject

These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bit 1 **RRFS**: Reject remote frames standard

- 0: Filter remote frames with 11-bit standard IDs
- 1: Reject all remote frames with 11-bit standard IDs

These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

Bit 0 **RRFE**: Reject remote frames extended

- 0: Filter remote frames with 29-bit standard IDs
- 1: Reject all remote frames with 29-bit standard IDs

These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

### 54.4.20 FDCAN extended ID and mask register (FDCAN_XIDAM)

**Address offset:** 0x0084

**Reset value:** 0x1FFF FFFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>EIDM[28:16]</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:29** Reserved, must be kept at reset value.

**Bits 28:0** **EIDM[28:0]**: Extended ID mask

For acceptance filtering of extended frames the Extended ID AND Mask is AND-ed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to 1 the mask is not active.

These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.
54.4.21 FDCAN high-priority message status register (FDCAN\_HPMS)

This register is updated every time a Message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0088

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLST**: Filter list

- Indicates the filter list of the matching filter element.
- 0: Standard filter list
- 1: Extended filter list

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:8 **FIDX[4:0]**: Filter index

- Index of matching filter element. Range is 0 to RXGFC[LSS] - 1 or RXGFC[LSE] - 1.

Bits 7:6 **MSI[1:0]**: Message storage indicator

- 00: No FIFO selected
- 01: FIFO overrun
- 10: Message stored in FIFO 0
- 11: Message stored in FIFO 1

Bits 5:3 Reserved, must be kept at reset value.

Bits 2:0 **BIDX[2:0]**: Buffer index

- Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

54.4.22 FDCAN Rx FIFO 0 status register (FDCAN\_RXF0S)

Address offset: 0x0090

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:26 Reserved, must be kept at reset value.
Bit 25 **RF0L**: Rx FIFO 0 message lost
This bit is a copy of interrupt flag IR[RF0L]. When IR[RF0L] is reset, this bit is also reset.
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 24 **F0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F0PI[1:0]**: Rx FIFO 0 put index
Rx FIFO 0 write index pointer, range 0 to 2.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **F0GI[1:0]**: Rx FIFO 0 get index
Rx FIFO 0 read index pointer, range 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F0FL[3:0]**: Rx FIFO 0 fill level
Number of elements stored in Rx FIFO 0, range 0 to 3.

### 54.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)

Address offset: 0x0094
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **F0AI[2:0]**: Rx FIFO 0 acknowledge index
After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This sets the Rx FIFO 0 get index RXF0S[F0GI] to F0AI + 1 and update the FIFO 0 fill level RXF0S[F0FL].

### 54.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)

Address offset: 0x0098
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

|                         | r   |     |     |     |     | r   |     | r   |     |     |     |     |     |     |     |

2568/3119   RM0481 Rev 1
54.4.25  FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)

Address offset: 0x009C

Reset value: 0x0000 0000

Bits 31:3  Reserved, must be kept at reset value.

Bits 2:0  F1AI[2:0]: Rx FIFO 1 acknowledge index
After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This sets the Rx FIFO 1 get index RXF1S[F1GI] to F1AI + 1 and update the FIFO 1 Fill Level RXF1S[F1FL].
54.4.26  FDCAN Tx buffer configuration register (FDCAN_TXBC)

Address offset: 0x00C0
Reset value: 0x0000 0000

| Bit 31:25  | Reserved, must be kept at reset value. |
| Bit 24    | **TFQM**: Tx FIFO/queue mode |
| 0: Tx FIFO operation |
| 1: Tx queue operation. |
| This is a protected write (P) bit, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1. |

| Bit 21    | **TFQF**: Tx FIFO/queue full |
| 0: Tx FIFO/queue not full |
| 1: Tx FIFO/queue full |

| Bit 19:18 | Reserved, must be kept at reset value. |
| Bit 17:16 | **TFQPI[1:0]**: Tx FIFO/queue put index |
| Tx FIFO/queue write index pointer, range 0 to 3 |

| Bit 15:10 | Reserved, must be kept at reset value. |
| Bit 9:8   | **TFGI[1:0]**: Tx FIFO get index |
| Tx FIFO read index pointer, range 0 to 3. Read as 0 when Tx queue operation is configured (TXBC.TFQM = 1) |

54.4.27  FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)

The Tx FIFO/Queue status is related to the pending Tx requests listed in register TXBRP. Therefore the effect of Add/Cancellation requests may be delayed due to a running Tx scan (TXBRP not yet updated).

Address offset: 0x00C4
Reset value: 0x0000 0003

| Bit 31:22 | Reserved, must be kept at reset value. |
| Bit 21    | **TFQF**: Tx FIFO/queue full |
| 0: Tx FIFO/queue not full |
| 1: Tx FIFO/queue full |

| Bit 19:18 | Reserved, must be kept at reset value. |
| Bit 17:16 | **TFQPI[1:0]**: Tx FIFO/queue put index |
| Tx FIFO/queue write index pointer, range 0 to 3 |

| Bit 15:10 | Reserved, must be kept at reset value. |
| Bit 9:8   | **TFGI[1:0]**: Tx FIFO get index |
| Tx FIFO read index pointer, range 0 to 3. Read as 0 when Tx queue operation is configured (TXBC.TFQM = 1) |
54.4.28 **FDCAN Tx buffer request pending register (FDCAN_TXBRP)**

Address offset: 0x00C8

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TRP[2:0]: Transmission request pending**

Each Tx buffer has its own transmission request pending bit. The bits are set via register TXBAR. The bits are reset after a requested transmission has completed or has been canceled via register TXBCR.

After a TXBRP bit has been set, a Tx scan is started to check for the pending Tx request with the highest priority (Tx buffer with lowest Message ID).

A cancellation request resets the corresponding transmission request pending bit of register TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset.

After a cancellation has been requested, a finished cancellation is signaled via TXBCF after successful transmission together with the corresponding TXBTO bit when the transmission has not yet been started at the point of cancellation when the transmission has been aborted due to lost arbitration when an error occurred during frame transmission

In DAR mode all transmissions are automatically canceled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending
1: Transmission request pending

**Note:** *TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this Add Request is canceled immediately, the corresponding TXBRP bit is reset.*
54.4.29  FDCAN Tx buffer add request register (FDCAN_TXBAR)

Address offset: 0x00CC
Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:3  Reserved, must be kept at reset value.

Bits 2:0  **AR[2:0]: Add request**

Each Tx buffer has its own add request bit. Writing a 1 sets the corresponding add request bit; writing a 0 has no impact. This enables the Host to set transmission requests for multiple Tx buffers with one write to TXBAR. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed.

0: No transmission request added
1: Transmission requested added.

**Note:** If an add request is applied for a Tx buffer with pending transmission request (corresponding TXBRP bit already set), the request is ignored.

54.4.30  FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)

Address offset: 0x00D0
Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:3  Reserved, must be kept at reset value.

Bits 2:0  **CR[2:0]: Cancellation request**

Each Tx buffer has its own cancellation request bit. Writing a 1 sets the corresponding CR bit; writing a 0 has no impact. This enables the Host to set cancellation requests for multiple Tx buffers with one write to TXBCR. The bits remain set until the corresponding TXBRP bit is reset.

0: No cancellation pending
1: Cancellation pending
## 54.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)

Address offset: 0x00D4  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:3 Reserved, must be kept at reset value. 

Bits 2:0 **TO[2:0]:** Transmission occurred.  
Each Tx buffer has its own TO bit. The bits are set when the corresponding TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register TXBAR.  
0: No transmission occurred  
1: Transmission occurred

## 54.4.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)

Address offset: 0x00D8  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:3 Reserved, must be kept at reset value. 

Bits 2:0 **CF[2:0]:** Cancellation finished  
Each Tx buffer has its own CF bit. The bits are set when the corresponding TXBRP bit is cleared after a cancellation was requested via TXBCR. In case the corresponding TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register TXBAR.  
0: No transmit buffer cancellation  
1: Transmit buffer cancellation finished
54.4.33  FDCAN Tx buffer transmission interrupt enable register  
(FDCAN_TXBTIE)  
Address offset: 0x00DC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0  **TIE[2:0]**: Transmission interrupt enable  
Each Tx buffer has its own TIE bit.  
0: Transmission interrupt disabled  
1: Transmission interrupt enabled

54.4.34  FDCAN Tx buffer cancellation finished interrupt enable register  
(FDCAN_TXBCIE)  
Address offset: 0x00E0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0  **CFIE[2:0]**: Cancellation finished interrupt enable.  
Each Tx buffer has its own CFIE bit.  
0: Cancellation finished interrupt disabled  
1: Cancellation finished interrupt enabled
54.4.35  **FDCAN Tx event FIFO status register (FDCAN_TXEFS)**

Address offset: 0x00E4  
Reset value: 0x0000 0000

| Bit 31:26 | Reserved, must be kept at reset value. |
| Bit 25   | TEFL: Tx event FIFO element lost  
This bit is a copy of interrupt flag IR[TEFL]. When IR[TEFL] is reset, this bit is also reset.  
0 No Tx event FIFO element lost  
1 Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0. |
| Bit 24   | EFF: Event FIFO full  
0: Tx event FIFO not full  
1: Tx event FIFO full |
| Bit 17:16 | EFPI[1:0]: Event FIFO put index  
Tx event FIFO write index pointer, range 0 to 3. |
| Bit 15:10 | Reserved, must be kept at reset value. |
| Bit 9:8  | EFGI[1:0]: Event FIFO get index  
Tx event FIFO read index pointer, range 0 to 3. |
| Bit 7:3  | Reserved, must be kept at reset value. |
| Bit 2:0  | EFFL[2:0]: Event FIFO fill level  
Number of elements stored in Tx event FIFO, range 0 to 3. |

54.4.36  **FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)**

Address offset: 0x00E8  
Reset value: 0x0000 0000

| Bit 31:2 | Reserved, must be kept at reset value. |
Bits 1:0 **EFAI[1:0]**: Event FIFO acknowledge index
After the Host has read an element or a sequence of elements from the Tx event FIFO, it has to write the index of the last element read from Tx event FIFO to EFAI. This sets the Tx event FIFO get index TXEFS[EFGI] to EFAI + 1 and updates the FIFO 0 fill level TXEFS[EFFL].

### 54.4.37 FDCAN CFG clock divider register (FDCAN_CKDIV)

**Address offset**: 0x0100  
**Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:4 Reserved, must be kept at reset value.

**Bits 3:0 PDIV[3:0]**: input clock divider
The APB clock can be divided prior to be used by the CAN sub system. The rate must be computed using the divider output clock.

- 0000: Divide by 1
- 0001: Divide by 2
- 0010: Divide by 4
- 0011: Divide by 6
- 0100: Divide by 8
- 0101: Divide by 10
- 0110: Divide by 12
- 0111: Divide by 14
- 1000: Divide by 16
- 1001: Divide by 18
- 1010: Divide by 20
- 1011: Divide by 22
- 1100: Divide by 24
- 1101: Divide by 26
- 1110: Divide by 28
- 1111: Divide by 30

These are protected write (P) bits, which means that write access by the bits is possible only when the bit 1 [CCE] and bit 0 [INIT] of CCCR register are set to 1.

### 54.4.38 FDCAN register map

**Table 589. FDCAN register map and reset values**

| Offset | Register | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

| Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
Table 589. FDCAN register map and reset values (1) (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset (hex)</th>
<th>Register (hex)</th>
<th>Register (hex)</th>
<th>Register (hex)</th>
<th>Register (hex)</th>
<th>Register (hex)</th>
<th>Register (hex)</th>
<th>Register (hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x004</td>
<td>FDCAN_ENDIAN</td>
<td>0x000</td>
<td>ETV[31:0]</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x008</td>
<td>Reserved</td>
<td>0x000</td>
<td>Reserved</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x00C</td>
<td>FDCAN_DBTP</td>
<td>0x000</td>
<td>DBRP[4:0]</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x010</td>
<td>FDCAN_TEST</td>
<td>0x000</td>
<td>DTSEG1[4:0]</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x014</td>
<td>FDCAN_RWD</td>
<td>0x000</td>
<td>DTSEG2[3:0]</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x018</td>
<td>FDCAN_NBRTP</td>
<td>0x000</td>
<td>DSJW[3:0]</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x01C</td>
<td>FDCAN_DBTP</td>
<td>0x000</td>
<td>DBRP[4:0]</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x020</td>
<td>FDCAN_TSCC</td>
<td>0x000</td>
<td>DTSEG1[4:0]</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x024</td>
<td>FDCAN_TSCV</td>
<td>0x000</td>
<td>DTSEG2[3:0]</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x028</td>
<td>FDCAN_TOCC</td>
<td>0x000</td>
<td>DSJW[3:0]</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x02C</td>
<td>FDCAN_TOCC</td>
<td>0x000</td>
<td>DBRP[4:0]</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x030-</td>
<td>Reserved</td>
<td>0x000</td>
<td>DTSEG1[4:0]</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x034</td>
<td>FDCAN_ECR</td>
<td>0x000</td>
<td>DTSEG2[3:0]</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x038</td>
<td>FDCAN_PSR</td>
<td>0x000</td>
<td>DSJW[3:0]</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x03C</td>
<td>FDCAN_TDCR</td>
<td>0x000</td>
<td>DBRP[4:0]</td>
<td>0x1</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
</tbody>
</table>

Notes:
1. TDC, DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] are 6-bit registers.
2. DSJW[3:0] is a 4-bit register.
3. RX, TX[1:0], BACK, and LACK are 4-bit registers.
4. TSC[15:0], TSCP[3:0], TSC[15:0], TOP[15:0], TOS[1:0], and ETOC are 16-bit registers.
5. DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] are 6-bit registers.
6. DSJW[3:0] is a 4-bit register.
7. RX, TX[1:0], BACK, and LACK are 4-bit registers.
8. TSC[15:0], TSCP[3:0], TSC[15:0], TOP[15:0], TOS[1:0], and ETOC are 16-bit registers.
9. DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] are 6-bit registers.
10. DSJW[3:0] is a 4-bit register.
11. RX, TX[1:0], BACK, and LACK are 4-bit registers.
12. TSC[15:0], TSCP[3:0], TSC[15:0], TOP[15:0], TOS[1:0], and ETOC are 16-bit registers.
13. DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] are 6-bit registers.
14. DSJW[3:0] is a 4-bit register.
15. RX, TX[1:0], BACK, and LACK are 4-bit registers.
16. TSC[15:0], TSCP[3:0], TSC[15:0], TOP[15:0], TOS[1:0], and ETOC are 16-bit registers.
17. DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] are 6-bit registers.
18. DSJW[3:0] is a 4-bit register.
19. RX, TX[1:0], BACK, and LACK are 4-bit registers.
20. TSC[15:0], TSCP[3:0], TSC[15:0], TOP[15:0], TOS[1:0], and ETOC are 16-bit registers.
21. DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] are 6-bit registers.
22. DSJW[3:0] is a 4-bit register.
23. RX, TX[1:0], BACK, and LACK are 4-bit registers.
24. TSC[15:0], TSCP[3:0], TSC[15:0], TOP[15:0], TOS[1:0], and ETOC are 16-bit registers.
25. DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] are 6-bit registers.
26. DSJW[3:0] is a 4-bit register.
27. RX, TX[1:0], BACK, and LACK are 4-bit registers.
28. TSC[15:0], TSCP[3:0], TSC[15:0], TOP[15:0], TOS[1:0], and ETOC are 16-bit registers.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0050</td>
<td>FDCAN_IR</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0054</td>
<td>FDCAN_IE</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x005B</td>
<td>FDCAN_ILS</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x005C</td>
<td>FDCAN_IIE</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x005E-0x007C</td>
<td>Reserved</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x0080</td>
<td>FDCAN_RXGFC</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0084</td>
<td>FDCAN_XIDAM</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0088</td>
<td>FDCAN_HPMS</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0090</td>
<td>FDCAN_RXF0S</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0094</td>
<td>FDCAN_RXF0A</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0098</td>
<td>FDCAN_RXF1S</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x009C</td>
<td>FDCAN_RXF1A</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x00A0-0x00BC</td>
<td>Reserved</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x00C0</td>
<td>FDCAN_TXBC</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>Reset value</td>
</tr>
</tbody>
</table>
Table 589. FDCAN register map and reset values (1) (continued)

| Offset | Register         | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00C4 | FDCAN _TXFQS     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00CB | FDCAN _TXBRP     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00CC | FDCAN _TXBAR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00D0 | FDCAN _TXBCR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00D4 | FDCAN _TXBTO     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00D8 | FDCAN _TXBCF     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00DC | FDCAN _TXBTIE    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00E0 | FDCAN _TXBCIE    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00E4 | FDCAN _TXEFS     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00E8 | FDCAN _TXEFA     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0100 | FDCAN _CKDIV     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

1. R = Read, S = Set on read, X = Reset on read, W = Write, P = Protected write, p = Protected set, C = Clear/preset on write.

Refer to Section 2.3 on page 111 for the register boundary addresses.
55 Universal serial bus full-speed host/device interface (USB)

55.1 Introduction
The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB2 bus.

USB suspend/resume are supported, which permits to stop the device clocks for low-power consumption.

55.2 USB main features
- USB specification version 2.0 full-speed compliant
- Supports both Host and Device modes
- Configurable number of endpoints from 1 to 8
- Dedicated packet buffer memory (SRAM) of 2048 bytes
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Isochronous transfers support
- Double-buffered bulk/isochronous endpoint/channel support
- USB Suspend/Resume operations
- Frame locked clock pulse generation
- USB 2.0 Link Power Management support (Device mode only)
- Battery Charging Specification Revision 1.2 support (Device mode only)
- USB connect / disconnect capability (controllable embedded pull-up resistor on USB_DP line)

55.3 USB implementation
Table 590 describes the USB implementation in the devices.

<table>
<thead>
<tr>
<th>Table 590. STM32H563/H573 and STM32H562 USB implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>USB features(1)</strong></td>
</tr>
<tr>
<td>Host mode</td>
</tr>
<tr>
<td>Number of endpoints</td>
</tr>
<tr>
<td>Size of dedicated packet buffer memory SRAM</td>
</tr>
<tr>
<td>Dedicated packet buffer memory SRAM access scheme</td>
</tr>
<tr>
<td>USB 2.0 Link Power Management (LPM) support in device</td>
</tr>
<tr>
<td>Battery Charging Detection (BCD) support for device</td>
</tr>
<tr>
<td>Embedded pull-up resistor on USB_DP line</td>
</tr>
</tbody>
</table>

1. X= supported
55.4 USB functional description

Figure 780 shows the block diagram of the USB peripheral.

Figure 780. USB peripheral block diagram

General description and Device mode functionality

The USB peripheral provides a USB-compliant connection between the function implemented by the microcontroller and an external USB function which can be a host PC but also a USB Device. Data transfer between the external USB host or device and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. This dedicated memory size is 2048 bytes, and up to 16 mono-directional or 8 bidirectional endpoints can be used. The USB peripheral interfaces with the external USB Host or Device, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.
Each endpoint/channel is associated with a buffer description block indicating where the endpoint/channel-related memory area is located, how large it is or how many bytes must be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint/channel is configured) takes place. The data buffered by the USB peripheral are loaded in an internal 16-bit register and memory access to the dedicated buffer is performed. When all the data have been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint/channel-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- which endpoint/channel has to be served,
- which type of transaction took place, if errors occurred (bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.).

Special support is offered to isochronous transfers and high throughput bulk transfers, implementing a double buffer usage, which permits to always have an available buffer for the USB peripheral while the microcontroller uses the other one.

A special bit THR512 in register USB_ISTR allows notification of 512 bytes being received in (or transmitted from) the buffer. This bit must be used for long ISO packets (from 512 to 1023 bytes) as it facilitates early start or read/write of data. In this way, the first 512 bytes can be handled by software while avoiding use of double buffer mode. This bit works when only one ISO endpoint is configured.

The unit can be placed in low-power mode (SUSPEND mode), by writing in the control register, whenever required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wake-up line to permit the system to immediately restart the normal clock generation and/or support direct clock start/stop.

**Host mode and specific functionality**

A single bit, HOST, in register USB_CNTR permits Host mode to be activated. Host mode functionality permits the USB to talk to a remote peripheral. Supported functionality is aligned to Device mode and uses the same register structures to manage the buffers. The same number of endpoints can be supported in Host mode, however in Host mode the terminology “channel” is preferred, as each channel is in reality a combination of the connected device and the endpoint on that device. The basic mechanisms for packet transmission and reception are the same as those supported in Device mode.

When operating in Host mode, the USB is in charge of the bus and in order to do this must issue transaction requests corresponding to active periodic and non-periodic endpoints. A host frame scheduler assures efficient use of the frame. Connection to hubs is supported. Connection to low speed devices is supported, both with a direct connection and through a hub.

Double-buffered mode, as previously described in Device mode, is also supported in Host mode, in both bulk and isochronous channels. The THR512 functionality is also supported (but as in Device mode) only for ISO traffic.

**Note:** Unlike in Device mode, where there is a detection of battery charging capability (in order to facilitate fast charging), there is no integrated support in Host mode to present battery.
charging capability (CDP or DCP cases in the standard), the host port is always presented as a default standard data port (SDP).

**Note:** For LPM (link power management) this feature is not supported in Host mode.

### 55.4.1 Description of USB blocks used in both Device and Host modes

The USB peripheral implements all the features related to USB interfacing, which include the following blocks:

- USB physical interface (USB PHY): this block is maintaining the electrical interface to an external USB host. It contains the differential analog transceiver itself, controllable embedded pull-up resistor (connected to USB_DP line) and support for battery charging detection (BCD), multiplexed on same USB_DP and USB_DM lines. The output enable control signal of the analog transceiver (active low) is provided externally on USB_NOE. It can be used to drive some activity LED or to provide information about the actual communication direction to some other circuitry.

- Serial interface engine (SIE): the functions of this block include: synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and uses the virtual buffers provided by the packet buffer interface for local data storage. This unit also generates signals according to USB peripheral events, such as start of frame (SOF), USB_Reset, data errors etc. and to endpoint related events like end of transmission or correct reception of a packet; these signals are then used to generate interrupts.

- Timer: this block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no traffic has been received for 3 ms.

- Packet buffer interface: this block manages the local memory implementing a set of buffers in a flexible way, both for transmission and reception. It can choose the proper buffer according to requests coming from the SIE and locate them in the memory addresses pointed by the endpoint/channel registers. It increments the address after each exchanged byte until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer to overrun the maximum capacity.

- Endpoint/channel-related registers: each endpoint/channel has an associated register containing the endpoint/channel type and its current status. For monodirectional/single-buffer endpoints, a single register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 monodirectional/single-buffer or up to 7 double-buffer endpoints in any combination. For example the USB peripheral can be programmed to have 4 double buffer endpoints and 8 single-buffer/mono-directional endpoints.

- Control registers: these are the registers containing information about the status of the whole USB peripheral and used to force some USB events, such as resume and power-down.

- Interrupt registers: these contain the interrupt masks and a record of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear the status of a pending interrupt.

**Note:** * Endpoint/channel 0 is always used for control transfer in single-buffer mode.

The USB peripheral is connected to the APB2 bus through an APB2 interface, containing the following blocks:

- Packet memory: this is the local memory that physically contains the packet buffers. It can be used by the packet buffer interface, which creates the data structure and can be
accessed directly by the application software. The size of the packet memory is 2048 bytes, structured as 512 words of 32 bits.

- Arbiter: this block accepts memory requests coming from the APB2 bus and from the USB interface. It resolves the conflicts by giving priority to APB2 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, while an USB transaction is happening. Multiword APB2 transfers of any length are also allowed by this scheme.

- Register mapper: this block collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 32-bit wide word set addressed by the APB2.

- APB2 wrapper: this provides an interface to the APB2 for the memory and register. It also maps the whole USB peripheral in the APB2 address space.

- Interrupt mapper: this block is used to select how the possible USB events can generate interrupts and map them to the NVIC.

55.4.2 Description of host frame scheduler (HFS) specific to Host mode

The host frame scheduler is the hardware machine in charge to submit host channel requests on the bus according to the USB priority order and bandwidth access rules.

Host channels are divided in two categories:
- Periodic channels: isochronous and interrupt traffic types. With guaranteed bandwith access.
- Non-periodic channels: bulk and control traffic types. With best effort service.

The host frame scheduler organizes the full-speed frame in 3 sequential windows
- Periodic service window
- Non-periodic service window
- Black security window

At the start of a new frame the host scheduler:
1. First considers all periodic channels which were active (STAT bits VALID) at the start of frame
2. Executes single round of service of periodic channels, the periodic service window, in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first
3. When the periodic round is finished, HFS closes the periodic service window and stops servicing periodic traffic even if some periodic channel was re-enabled or some new channel was enabled after the SOF.
4. Starts servicing all non-periodic channels which are currently active (STAT bits VALID) in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first.
5. Executes multiple round-robin service cycles of non-periodic channels until almost the end of frame
6. Non periodic traffic can be requested at any time and is serviced by HFS with best effort latency, with the exception of a black security window at the end of the frame where new injected requests are directly postponed to the next frame to avoid babbles. This is also true for pending transactions which have not been serviced ahead of the security window.
55.5 Programming considerations for Device and Host modes

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

55.5.1 Generic USB Device programming

This part describes the main tasks required of the application software in order to obtain USB compliant behavior. The actions related to the most general USB events are taken into account and paragraphs are dedicated to the special cases of double-buffered endpoints and isochronous transfers. Apart from system reset, an action is always initiated by the USB peripheral, driven by one of the USB events described below.

55.5.2 System and power-on reset

Upon system and power-on reset, the first operation the application software must perform is to provide all required clock signals to the USB peripheral and subsequently de-assert its reset signal so to be able to access its registers. The whole initialization sequence is hereafter described.

As a first step application software needs to activate register macrocell clock and de-assert macrocell specific reset signal using related control bits provided by device clock management logic.

After that, the analog part of the device related to the USB transceiver must be switched on using the PDWN bit in CNTR register, which requires a special handling. This bit is intended to switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time (t\text{STARTUP} specified in the datasheet) during which the behavior of the USB transceiver is not defined. It is thus necessary to wait this time, after setting the PDWN bit in the CNTR register, before removing the reset condition on the USB part (by clearing the USBRST bit in the CNTR register). Clearing the ISTR register removes any spurious pending interrupt before any other macrocell operation is enabled.

At system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and data transfers. All registers not specific to any endpoint/channel must be initialized according to the needs of application software (choice of enabled interrupts, chosen address of packet buffers, etc.). Then the process continues as for the USB reset case (see further paragraph).

USB bus reset (RST_DCON interrupt) in Device mode

When this event occurs, the USB peripheral is put in the same conditions it is left by the system reset after the initialization described in the previous paragraph: communication is disabled in all endpoint registers (the USB peripheral does not respond to any packet). As a response to the USB reset event, the USB function must be enabled, having as USB address 0, implementing only the default control endpoint (endpoint address is 0 too). This is accomplished by setting the enable function (EF) bit of the USB_DADDR register and initializing the CHEP0R register and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADD[6:0] bits of the USB_DADDR register, and configures any other necessary endpoint.
When a RST_DCON interrupt is received, the application software is responsible to enable again the default endpoint of USB function 0 within 10 ms from the end of the reset sequence which triggered the interrupt.

**USB bus reset in Host mode**

In Host mode a bus reset is activated by setting the USBRST bit of the USB_CNTR register. It must subsequently be cleared by software once the minimum active reset time from the standard has been respected.

**Structure and usage of packet buffers**

Each bidirectional endpoint may receive or transmit data over the bus. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface block, which delivers a memory access request and waits for its acknowledgment. Since the packet buffer memory has also to be accessed by the microcontroller, an arbitration logic takes care of the access conflicts, using half APB2 cycle for microcontroller access and the remaining half for the USB peripheral access. In this way, both agents can operate as if the packet memory would be a dual-port SRAM, without being aware of any conflict even when the microcontroller is performing back-to-back accesses. The USB peripheral logic uses a dedicated clock. The frequency of this dedicated clock is fixed by the requirements of the USB standard at 48 MHz, and this can be different from the clock used for the interface to the APB2 bus. Different clock configurations are possible where the APB2 clock frequency can be higher or lower than the USB peripheral one.

*Note:* Due to USB data rate and packet memory interface requirements, the APB2 clock must have a minimum frequency of 12 MHz to avoid data overrun/underrun problems.

Each endpoint is associated with two packet buffers (usually one for transmission and the other one for reception). Buffers can be placed anywhere inside the packet memory because their location and size is specified in a buffer description table, which is also located in the packet memory. Each table entry is associated to an endpoint register and it is composed of two 32-bit words so that table start address must always be aligned to an 8-byte boundary. Buffer descriptor table entries are described in Section 55.6.2: Buffer descriptor table. If an endpoint is unidirectional and it is neither an isochronous nor a double-buffered bulk, only one packet buffer is required (the one related to the supported transfer direction). Other table locations related to unsupported transfer directions or unused endpoints, are available to the user. Isochronous and double-buffered bulk endpoints have special handling of packet buffers (Refer to Section 55.5.5: Isochronous transfers in Device mode and Section 55.5.3: Double-buffered endpoints and usage in Device mode respectively). The relationship between buffer description table entries and packet buffer areas is depicted in Figure 781.

For Host mode different sections explain the buffer usage model, notably Section 55.5.6: Isochronous transfers in Host mode and Section 55.5.4: Double buffered channels: usage in Host mode.
Each packet buffer is used either during reception or transmission starting from the bottom. The USB peripheral never changes the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data is copied to the memory only up to the last available location.

**Endpoint initialization**

The first step to initialize an endpoint is to write appropriate values to the ADDRn_TX/ADDRn_RX fields in the CHEP_TXBD_n and CHEP_RXBD_n registers (in SRAM) so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The UTYPE bits in the USB_CHEPnR register must be set according to the endpoint type, eventually using the EPKIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STATTX bits in the USB_CHEPnR register and COUNTn_TX must be initialized. For reception, STATRX bits must be set to enable reception and COUNTn_RX must be written with the allocated buffer size using the BLSIZE and NUM_BLOCK fields. Unidirectional endpoints, except isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, register USB_CHEPnR and locations ADDRn_TX/ADDRn_RX, COUNTn_TX/COUNTn_RX (respectively), must not be modified by the application software, as the hardware can...
change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

Data transmission in Device mode (IN packets)

When receiving an IN token packet, if the received address matches a configured and valid endpoint, the USB peripheral accesses the contents of CHEP_TXBD_n (fields ADDRn_TX and COUNTn_TX) inside the buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16-bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first byte to be transmitted (refer to Structure and usage of packet buffers on page 2586) and the USB peripheral starts sending a DATA0 or DATA1 PID according to USB_CHEPnR bit DTOGTX. When the PID is completed, the first byte, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STATTX bits in the USB_CHEPnR register.

The ADDRn_TX field in the internal register CHEP_TXBD_n is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each half-word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRn_TX for COUNTn_TX/4 words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last half-word accessed is used.

On receiving the ACK receipt by the host, the USB_CHEPnR register is updated in the following way: DTOGTX bit is toggled, the endpoint is made invalid by setting STATTX = 10 (NAK) and bit VTTX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB_ISTR register. Servicing of the VTTX event starts, clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTn_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STATTX to 11 (VALID) to re-enable transmission. While the STATTX bits are equal to 10 (NAK), any IN request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second IN transaction addressed to the same endpoint immediately following the one which triggered the CTR interrupt.

Data transmission in Host mode (OUT packets)

Data transmission in Host mode follows the same general principles as Device mode. The main differences are due to the protocol. For example the host initiates the transmission whereas the device responds to the incoming token.

ADDRn_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents of an OUT packet are then written to that address in the packet memory and COUNTn_TX must be updated (when necessary) to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATTX must be set to 11 (VALID) in order to trigger the transmit. The transmission is then scheduled by the HFS.

After a successful transmission the CTR interrupt (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the
indicated channel, the STATTX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATTX is now in NAK. In the case of a STALL response, STATTX is in STALL. In this last case, the bus must be reset.

On receiving the ACK receipt by the device, the USB_CHEPnR register is updated in the following way: DTOGTX bit is toggled.

An error condition is signaled via the bits VTTX and ERR_TX in the case of:
- No handshake being received in time
- False EOP
- Bit stuffing error
- Invalid handshake PID

**Data reception in Device mode (OUT and SETUP packets)**

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following paragraph about control transfers. When receiving an OUT/SETUP PID, if the address matches a valid endpoint, the USB peripheral accesses the contents of the ADDRn_RX and COUNTn_RX fields inside the buffer descriptor table entry related to the addressed endpoint. The content of the ADDRn_RX field is stored directly in its internal register ADDR. Internal register COUNT is now reset and the values of BLSIZE and NUM_BLOCK bit fields, which are read within USB_CHEP_RXBD_n content, are used to initialize BUF_COUNT, an internal 16-bit counter, which is used to check the buffer overrun condition (all these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in half-words (the first byte received is stored as least significant byte) and then transferred to the packet buffer starting from the address contained in the internal ADDR register while BUF_COUNT is decremented and COUNT is incremented at each byte transfer. When the end of DATA packet is detected, the correctness of the received CRC is tested and only if no errors occurred during the reception, an ACK handshake packet is sent back to the transmitting host.

In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied in the packet memory buffer, at least until the error detection point, but the ACK packet is not sent and the ERR bit in USB_ISTR register is set. However, there is usually no software action required in this case: the USB peripheral recovers from reception errors and remains ready for the next transaction to come. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to bits STATRX in the USB_CHEPnR register, and no data is written in the reception memory buffers.

Reception memory buffer locations are written starting from the address contained in the ADDRn_RX for a number of bytes corresponding to the received data packet length, or up to the last allocated memory location, as defined by BLSIZE and NUM_BLOCK, whichever comes first. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

When the transaction is completed correctly, by sending the ACK handshake packet, the internal COUNT register is copied back in the COUNTn_RX location inside the buffer description table entry, leaving unaffected BLSIZE and NUM_BLOCK fields, which normally
do not require to be re-written, and the USB_CHEPnR register is updated in the following way: DTOGRX bit is toggled, the endpoint is made invalid by setting STATRX = 10 (NAK) and bit VTRX is set. If the transaction has failed due to errors or buffer overrun condition, none of the previously listed actions take place. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USBISTR register. The VTRX event is serviced by first determining the transaction type (SETUP bit in the USB_CHEPnR register); the application software must clear the interrupt flag bit and get the number of received bytes reading the COUNTn_RX location inside the buffer description table entry related to the endpoint being processed.

After the received data is processed, the application software must set the STATRX bits to 11 (VALID) in the USB_CHEPnR, enabling further transactions. While the STATRX bits are equal to 10 (NAK), any OUT request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second OUT transaction addressed to the same endpoint following immediately the one which triggered the CTR interrupt.

Data reception in Host mode (IN packets)

Data reception in Host mode follows the same general principles as Device mode. The main differences are again due to the protocol. In the device, data can be received or not, depending on readiness after previous operations, whereas the host only requests receive data when it is ready and able to store them.

ADDRn_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents received in the data phase response to the IN token packet are then written to that address in the packet memory and COUNTn_TX gets updated by hardware during this process to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATRX must be set to VALID in order to trigger the reception. The reception is then scheduled by the HFS.

After a successful reception the interrupt CTR (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the indicated channel, the STATRX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATRX now is in NAK. In the case of a STALL response, STATRX is in STALL. In this last case, the bus must be reset. During an IN packet an error condition is signaled via the bits VTRX and ERR_RX in case of:

- False EOP
- Bit stuffing error
- Wrong CRC

Control transfers in Device mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only and are very similar to OUT ones (data reception) except that the values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 1 and 0 respectively, to initialize the control transfer, and both STATTX and STATRX are set to 10 (NAK) to let software decide if subsequent transactions must be IN or OUT depending on the SETUP contents. A control endpoint must check SETUP bit in the USB_CHEPnR register at each VTRX event to distinguish normal OUT transactions from SETUP ones. A USB Device can determine the number and direction of data stages by interpreting the data transferred in the SETUP stage, and is
required to STALL the transaction in the case of errors. To do so, at all data stages before the last, the unused direction must be set to STALL, so that, if the host reverses the transfer direction too soon, it gets a STALL as a status stage.

While enabling the last data stage, the opposite direction must be set to NAK, so that, if the host reverses the transfer direction (to perform the status stage) immediately, it is kept waiting for the completion of the control operation. If the control operation completes successfully, the software changes NAK to VALID, otherwise to STALL. At the same time, if the status stage is an OUT, the STATUS_OUT (EPKIND in the USB_CHEPnR register) bit must be set, so that an error is generated if a status transaction is performed with non-zero data. When the status transaction is serviced, the application clears the STATUS_OUT bit and sets STATRX to VALID (to accept a new command) and STATTX to NAK (to delay a possible status stage immediately following the next setup).

Since the USB specification states that a SETUP packet cannot be answered with a handshake different from ACK, eventually aborting a previously issued command to start the new one, the USB logic does not permit a control endpoint to answer with a NAK or STALL packet to a SETUP token received from the host.

When the STATRX bits are set to 01 (STALL) or 10 (NAK) and a SETUP token is received, the USB accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued VTRX request not yet acknowledged by the application (for example VTRX bit is still set from a previously completed reception), the USB discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the VTRX interrupt.

Control transfers in Host mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only. A control endpoint must set the SETUP bit in the USB_CHEPnR register. The values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 0. Depending on whether it is a control write or control read then STATTX or STATRX are set to 11 (ACTIVE) in order to trigger the control transfer via the host frame scheduler.

On receiving a CTR interrupt the channel (device address and endpoint) can be determined by examining IDN and DIR bits. Devices are expected to NAK every control unless the packet is corrupted in which case they do not acknowledge. The situation is reflected in the value of STATTX.

In the case of an error condition the ERR bit gets set. One possible case is where a CRC error is seen at the device, in this case no ACK is returned to the host. The host sees no ACK and after an appropriate delay this generates a timeout error with ERR_TX set (which can generate an interrupt).
55.5.3 Double-buffered endpoints and usage in Device mode

All different endpoint types defined by the USB standard represent different traffic models, and describe the typical requirements of different kind of data transfer operations. When large portions of data are to be transferred between the host PC and the USB function, the bulk endpoint type is the most suited model. This is because the host schedules bulk transactions so as to fill all the available bandwidth in the frame, maximizing the actual transfer rate as long as the USB function is ready to handle a bulk transaction addressed to it. If the USB function is still busy with the previous transaction when the next one arrives, it answers with a NAK handshake and the host PC issues the same transaction again until the USB function is ready to handle it, reducing the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called ‘double-buffering’ can be used with bulk endpoints.

When ‘double-buffering’ is activated, data toggle sequencing is used to select, which buffer is to be used by the USB peripheral to perform the required data transfers, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to always have a complete buffer to be used by the application, while the USB peripheral fills the other one. For example, during an OUT transaction directed to a ‘reception’ double-buffered bulk endpoint, while one buffer is being filled with new data coming from the USB host, the other one is available for the microcontroller software usage (the same would happen with a ‘transmission’ double-buffered bulk endpoint and an IN transaction).

Since the swapped buffer management requires the usage of all 4 buffer description table locations hosting the address pointer and the length of the allocated memory buffers, the USB_CHEPnR registers used to implement double-buffered bulk endpoints are forced to be used as unidirectional ones. Therefore, only one STAT bit pair must be set at a value different from 00 (DISABLED): STATRX if the double-buffered bulk endpoint is enabled for reception, STATTX if the double-buffered bulk endpoint is enabled for transmission. In case it is required to have double-buffered bulk endpoints enabled both for reception and transmission, two USB_CHEPnR registers must be used.

To exploit the double-buffering feature and reach the highest possible transfer rate, the endpoint flow control structure, described in previous chapters, has to be modified, in order to switch the endpoint status to NAK only when a buffer conflict occurs between the USB peripheral and application software, instead of doing it at the end of each successful transaction. The memory buffer which is currently being used by the USB peripheral is defined by the DTOG bit related to the endpoint direction: DTOGRX (bit 14 of USB_CHEPnR register) for ‘reception’ double-buffered bulk endpoints or DTOGTX (bit 6 of USB_CHEPnR register) for ‘transmission’ double-buffered bulk endpoints. To implement the new flow control scheme, the USB peripheral must know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USB_CHEPnR register, there are two DTOG bits but only one is used by USB peripheral for data and buffer sequencing (due to the unidirectional constraint required by double-buffering feature) the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called SW_BUF. In the following table the correspondence between USB_CHEPnR register bits and DTOG/SW_BUF definition is explained, for the cases of ‘transmission’ and ‘reception’ double-buffered bulk endpoints.
The memory buffer which is currently being used by the USB peripheral is defined by DTOG buffer flag, while the buffer currently in use by application software is identified by SW_BUF buffer flag. The relationship between the buffer flag value and the used packet buffer is the same in both cases, and it is listed in the following table.

Table 591. Double-buffering buffer flag definition

<table>
<thead>
<tr>
<th>Buffer flag</th>
<th>‘Transmission’ endpoint</th>
<th>‘Reception’ endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTOG</td>
<td>DTOGTX (USB_CHEPnR bit 6)</td>
<td>DTOGRX (USB_CHEPnR bit 14)</td>
</tr>
<tr>
<td>SW_BUF</td>
<td>USB_CHEPnR bit 14</td>
<td>USB_CHEPnR bit 6</td>
</tr>
</tbody>
</table>

Double-buffering feature for a bulk endpoint is activated by:
- Writing UTYPE bit field at 00 in its USB_CHEPnR register, to define the endpoint as a bulk, and
- Setting EPKIND bit at 1 (DBL_BUF), in the same register.

Table 592. Bulk double-buffering memory buffers usage (Device mode)

<table>
<thead>
<tr>
<th>Endpoint type</th>
<th>DTOG</th>
<th>SW_BUF</th>
<th>Packet buffer used by USB peripheral</th>
<th>Packet buffer used by Application Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit (IN)</td>
<td>0</td>
<td>1</td>
<td>USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX)</td>
<td>USB_CHEP_RTXXBD_0 (ADDR_TX / COUNT_TX)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>USB_CHEP_RTXXBD_0 (ADDR_TX / COUNT_TX)</td>
<td>USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>None (1)</td>
<td>USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>None (1)</td>
<td>USB_CHEP_RTXXBD_0 (ADDR_TX / COUNT_TX)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td>Receive (OUT)</td>
<td>0</td>
<td>1</td>
<td>USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX)</td>
<td>USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX)</td>
<td>USB_CHEP_RTXXBD_0 (ADDR_RX / COUNT_RX)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>None (1)</td>
<td>USB_CHEP_RTXXBD_0 (ADDR_RX / COUNT_RX)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>None (1)</td>
<td>USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
</tbody>
</table>

1. Endpoint in NAK Status.
The application software is responsible for DTOG and SW_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL_BUF remain set. At the end of each transaction the VTRX or VTTX bit of the addressed endpoint USB_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_CHEPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after DBL_BUF setting, STAT bit pair is not affected by the transaction termination and its value remains 11 (VALID). However, as the token packet of a new transaction is received, the actual endpoint status is masked as 10 (NAK) when a buffer conflict between the USB peripheral and the application software is detected (this condition is identified by DTOG and SW_BUF having the same value, see Table 592 on page 2593). The application software responds to the CTR event notification by clearing the interrupt flag and starting any required handling of the completed transaction. When the application packet buffer usage is over, the software toggles the SW_BUF bit, writing 1 to it, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is limited only by the application elaboration time of a transaction data: if the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions due to flow control takes place and the actual transfer rate is limited only by the host PC.

The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from 11 (VALID) into the STAT bit pair of the related USB_CHEPnR register. In this case, the USB peripheral always uses the programmed endpoint status, regardless of the buffer usage condition.

55.5.4 Double buffered channels: usage in Host mode

In Host mode the underlying transmit and receive methods for double buffered channels are the same as those described for Device mode.

Similar to the Device mode table, a new table below Table 593: Bulk double-buffering memory buffers usage (Host mode) shows the programming settings for OUT and IN tokens.
55.5.5 Isochronous transfers in Device mode

The USB standard supports full speed peripherals requiring a fixed and accurate data production/consume frequency, defining this kind of traffic as ‘isochronous’. Typical examples of this data are: audio samples, compressed video streams, and in general any sort of sampled data having strict requirements for the accuracy of delivered frequency. When an endpoint is defined to be ‘isochronous’ during the enumeration phase, the host allocates in the frame the required bandwidth and delivers exactly one IN or OUT packet each frame, depending on endpoint direction. To limit the bandwidth requirements, no retransmission of failed transactions is possible for isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase and no ACK packet is expected or sent after the data packet. For the same reason, isochronous transfers do not support data toggle sequencing and always use DATA0 PID to start any data packet.

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID), any other value produces results not compliant to USB standard. Isochronous endpoints implement double-buffering
to ease application software development, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to have always a complete buffer to be used by the application, while the USB peripheral fills the other.

The memory buffer which is currently used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTOGRX for ‘reception’ isochronous endpoints, DTOGTX for ‘transmission’ isochronous endpoints, both in the related USB_CHEPnR register) according to Table 594.

### Table 594. Isochronous memory buffers usage

<table>
<thead>
<tr>
<th>Endpoint Type</th>
<th>DTOG bit value</th>
<th>Packet buffer used by the USB peripheral</th>
<th>Packet buffer used by the application software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit (IN)</td>
<td>0</td>
<td>USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX)</td>
<td>USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX)</td>
<td>USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX)</td>
</tr>
<tr>
<td>Receive (OUT)</td>
<td>0</td>
<td>USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX)</td>
<td>USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX)</td>
<td>USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX)</td>
</tr>
</tbody>
</table>

As it happens with double-buffered bulk endpoints, the USB_CHEPnR registers used to implement isochronous endpoints are forced to be used as unidirectional ones. In case it is required to have isochronous endpoints enabled both for reception and transmission, two USB_CHEPnR registers must be used.

The application software is responsible for the DTOG bit initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the VTRX or VTTX bit of the addressed endpoint USB_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_CHEPnR register is hardware toggled making buffer swapping completely software independent. STAT bit pair is not affected by transaction completion; since no flow control is possible for isochronous transfers due to the lack of handshake phase, the endpoint remains always 11 (VALID). CRC errors or buffer-overrun conditions occurring during isochronous OUT transfers are anyway considered as correct transactions and they always trigger a VTRX event. However, CRC errors set the ERR bit in the USB_ISTR register anyway, in order to notify the software of the possible data corruption.

### 55.5.6 Isochronous transfers in Host mode

From the host point of view isochronous packets are issued or requested one by frame by the host frame scheduler. There is no NAK/ACK protocol and no resend of data or token.
The mechanism is based on a table very similar to that for Device mode. See Table 595 below to understand the relationship between the DTOG bit buffers and the buffer usage.

### Table 595. Isochronous memory buffers usage

<table>
<thead>
<tr>
<th>Endpoint Type</th>
<th>DTOG bit value</th>
<th>Packet buffer used by the USB peripheral</th>
<th>Packet buffer used by the application software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit (OUT) 0</td>
<td>USB_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX)</td>
<td>USB_CHEP_RTXXB0D_0 (ADDRn_TX / COUNTn_TX)</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
<td></td>
</tr>
<tr>
<td>Transmit (OUT) 1</td>
<td>USB_CHEP_RTXXB0D_0 (ADDRn_TX / COUNTn_TX)</td>
<td>USB_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX)</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
<td></td>
</tr>
<tr>
<td>Receive (IN) 0</td>
<td>USB_CHEP_RTXXB0D_0 (ADDRn_RX / COUNTn_RX)</td>
<td>USB_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX)</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
<td></td>
</tr>
<tr>
<td>Receive (IN) 1</td>
<td>USB_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX)</td>
<td>USB_CHEP_RXRXBD_0 (ADDRn_RX / COUNTn_RX)</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
<td></td>
</tr>
</tbody>
</table>

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID).

Just as in Device mode, the mechanism allows automatic toggle of the DTOG bit. Note that in Host mode, at the same time as this toggle, the STATTX or STATRX of the completed buffer is automatically set to DISABLED, permitting the future buffer to be accessed before re-enabling it by setting it to 11 (VALID).

### 55.5.7 Suspend/resume events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus must not be greater than 2.5 mA. This requirement is of fundamental importance for bus-powered devices, while self-powered devices are not required to comply to this strict power consumption constraint. In suspend mode, the host PC sends the notification by not sending any traffic on the USB bus for more than 3 ms: since a SOF packet must be sent every 1 ms during normal operations, the USB peripheral detects the lack of 3 consecutive SOF packets as a suspend request from the host PC and set the SUSP bit to 1 in USB_ISTR register, causing an interrupt if enabled.

Once the device is suspended, its normal operation can be restored by a so called RESUME sequence, which can be started from the host PC or directly from the peripheral itself, but it is always terminated by the host PC. The suspended USB peripheral must be anyway able to detect a RESET sequence, reacting to this event as a normal USB reset event.

The actual procedure used to suspend the USB peripheral is device dependent since according to the device composition, different actions may be required to reduce the total consumption.
A brief description of a typical suspend procedure is provided below, focused on the USB-related aspects of the application software routine responding to the SUSP notification of the USB peripheral:

1. Set the SUSPEN bit in the USB_CNTR register to 1. This action activates the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSP interrupts being issued while the USB is suspended.

2. Remove or reduce any static power consumption in blocks different from the USB peripheral.

3. Set SUSPRDY bit in USB_CNTR register to 1 to remove static power consumption in the analog USB transceivers but keeping them able to detect resume activity.

4. Optionally turn off external oscillator and device PLL to stop any activity inside the device.

When an USB event occurs while the device is in SUSPEND mode, the RESUME procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular care must be taken to insure that this process does not take more than 10 ms when the wakening event is an USB reset sequence (see “Universal Serial Bus Specification” for more details). The start of a resume or reset sequence, while the USB peripheral is suspended, clears the SUSPRDY bit in USB_CNTR register asynchronously. Even if this event can trigger a WKUP interrupt if enabled, the use of an interrupt response routine must be carefully evaluated because of the long latency due to system clock restart; to have the shorter latency before re-activating the nominal clock it is suggested to put the resume procedure just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. To prevent ESD discharges or any other kind of noise from waking-up the system (the exit from suspend mode is an asynchronous event), a suitable analog filter on data line status is activated during suspend; the filter width is about 70 ns.

The following is a list of actions a resume procedure must address:

1. Optionally turn on external oscillator and/or device PLL.

2. Clear SUSPEN bit of USB_CNTR register.

3. If the resume triggering event has to be identified, bits RXDP and RXDM in the USB_FNR register can be used according to Table 596, which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected monitoring the status of the above mentioned bits by checking when they reach the “10” configuration, which represent the idle bus state; moreover at the end of a reset sequence the RST_DCON bit in USB_ISTR register is set to 1, issuing an interrupt if enabled, which must be handled as usual.
A device may require to exit from suspend mode as an answer to particular events not directly related to the USB protocol (for example a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the L2RES bit in the USB_CNTR register to 1 and resetting it to 0 after an interval between 1 ms and 15 ms (this interval can be timed using ESOF interrupts, occurring with a 1 ms period when the system clock is running at nominal frequency). Once the L2RES bit is clear, the resume sequence is completed by the host PC and its end can be monitored again using the RXDP and RXDM bits in the USB_FNR register.

*Note:* The L2RES bit must be anyway used only after the USB peripheral has been put in suspend mode, setting the SUSPEN bit in USB_CNTR register to 1.

**Suspend and resume in Host mode**

The basics of the suspend and resume mechanism has been described in the previous section.

From the host stand-point, suspend is entered by writing the SUSPEN bit in USB_CNTR. When suspend entry is confirmed, SUSPRDY (also in USB_CNTR) is set.

Once in suspend, and when the application want to resume the bus, this can be done by setting the L2RES bit in USB_CNTR to 1.

Below in Table 597, the different actions recommended after a wake-up event are indicated. According to the different line states after a wake-up event, the interpretation of the event and the suggested behavior are shown. Note that, this table here is somewhat expanded when compared to the previously shown device table, as the host may encounter both full speed and low speed devices which use different line states for both suspend and resume.

<table>
<thead>
<tr>
<th>[RXDP,RXDM] status</th>
<th>Wake-up event</th>
<th>Required resume software action</th>
</tr>
</thead>
<tbody>
<tr>
<td>“00”</td>
<td>Root reset</td>
<td>None</td>
</tr>
<tr>
<td>“10”</td>
<td>None (noise on bus)</td>
<td>Go back in Suspend mode</td>
</tr>
<tr>
<td>“01”</td>
<td>Root resume</td>
<td>None</td>
</tr>
<tr>
<td>“11”</td>
<td>Not allowed (noise on bus)</td>
<td>Go back in Suspend mode</td>
</tr>
</tbody>
</table>
Table 597. Resume event detection for host

<table>
<thead>
<tr>
<th>[RXDP,RXDM] status</th>
<th>Wake-up event</th>
<th>Required resume software action</th>
</tr>
</thead>
<tbody>
<tr>
<td>“00”</td>
<td>Not allowed (noise on bus)</td>
<td>Go back in Suspend mode</td>
</tr>
<tr>
<td>“10”</td>
<td>Full speed capable device: Not allowed (noise on bus) Low speed device: Device remote wake-up resume</td>
<td>None</td>
</tr>
<tr>
<td>“01”</td>
<td>Full speed capable device: Device remote wake-up resume Low speed device: Not allowed (noise on bus)</td>
<td>None</td>
</tr>
<tr>
<td>“11”</td>
<td>Not allowed (noise on bus)</td>
<td>Go back in Suspend mode</td>
</tr>
</tbody>
</table>
55.6 USB and USB SRAM registers

The USB peripheral registers can be divided into the following groups:
- Common registers: interrupt and control registers
- Endpoint/channel registers: endpoint/channel configuration and status

The USB SRAM registers cover:
- Buffer descriptor table: location of packet memory used to locate data buffers (see Section 2.3: Memory organization to find USB SRAM base address).

All register addresses are expressed as offsets with respect to the USB peripheral registers base address, except the buffer descriptor table locations, which starts at the USB SRAM base address.

Refer to Section 1.2 on page 101 for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

55.6.1 Common registers

These registers affect the general behavior of the USB peripheral defining operating mode, interrupt handling, device address and giving access to the current frame number updated by the host PC.

USB control register (USB_CNTR)

Address offset: 0x40
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| CTRM   | PMA | OVRM | ERRM | WKPUP | SUSP | M | RST_D | CONM | SOFM | ESOF | M | LREQ | M | Res. | LREQ | S | LREQ | S | SUSP | PEN | SUSP | RDY | PDWN | USB | RST |
|--------|-----|------|------|-------|------|---|-------|-----|------|-----|---|-----|---|----|-----|---|-----|---|-----|----|------|----|-----|----|
| rw     | rw  | rw   | rw   | rw    | rw   | rw| rw    | rw  | rw   | rw  | rw| rw   | rw| rw  | rw  | rw| rw   | rw| rw   | rw| rw   | rw| rw   | rw| rw   |

Bit 31 HOST: HOST mode
HOST bit selects between host or device USB mode of operation. It must be set before enabling the USB peripheral by the function enable bit.
0: USB Device function
1: USB host function

Bits 30:18 Reserved, must be kept at reset value.

Bit 17 DDISCM: Device disconnection mask
- Host mode
  0: Device disconnection interrupt disabled
  1: Device disconnection interrupt enabled

Bit 16 THR512M: 512 byte threshold interrupt mask
  0: 512 byte threshold interrupt disabled
  1: 512 byte threshold interrupt enabled
Bit 15 CTRM: Correct transfer interrupt mask
  0: Correct transfer (CTR) interrupt disabled.
  1: CTR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 14 PMAOVRM: Packet memory area over / underrun interrupt mask
  0: PMAOVR interrupt disabled.
  1: PMAOVR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 13 ERRM: Error interrupt mask
  0: ERR interrupt disabled.
  1: ERR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 12 WKUPM: Wake-up interrupt mask
  0: WKUP interrupt disabled.
  1: WKUP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 11 SUSPM: Suspend mode interrupt mask
  0: Suspend mode request (SUSP) interrupt disabled.
  1: SUSP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 10 RST_DCONM: USB reset request (Device mode) or device connect/disconnect (Host mode) interrupt mask
  0: RESET interrupt disabled.
  1: RESET interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 9 SOFM: Start of frame interrupt mask
  0: SOF interrupt disabled.
  1: SOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 8 ESOFM: Expected start of frame interrupt mask
  0: Expected start of frame (ESOF) interrupt disabled.
  1: ESOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 7 L1REQM: LPM L1 state request interrupt mask
  0: LPM L1 state request (L1REQ) interrupt disabled.
  1: L1REQ interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 6 Reserved, must be kept at reset value.

Bit 5 L1RES: L1 remote wake-up / resume driver
  – Device mode
    Software sets this bit to send a LPM L1 50 μs remote wake-up signaling to the host. After the signaling ends, this bit is cleared by hardware.
    0: No effect
    1: Send 50 μs remote-wake-up signaling to host
Bit 4 **L2RES**: L2 remote wake-up / resume driver

- **Device mode**
  
  The microcontroller can set this bit to send remote wake-up signaling to the host. It must be activated, according to USB specifications, for no less than 1 ms and no more than 15 ms after which the host PC is ready to drive the resume sequence up to its end.

- **Host mode**
  
  Software sets this bit to send resume signaling to the device.
  
  Software clears this bit to send end of resume to device and restart SOF generation.
  
  In the context of remote wake up, this bit is to be set following the WAKEUP interrupt.
  
  0: No effect
  
  1: Send L2 resume signaling to device

Bit 3 **SUSPEN**: Suspend state enable

- **Condition: Device mode**
  
  Software can set this bit when the SUSP interrupt is received, which is issued when no traffic is received by the USB peripheral for 3 ms. Software can also set this bit when the L1REQ interrupt is received with positive acknowledge sent.
  
  As soon as the suspend state is propagated internally all device activity is stopped, USB clock is gated, USB transceiver is set into low power mode and the SUSPRDY bit is set by hardware. In the case that device application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the microcontroller to stop mode, as in the case of bus powered device application, it must first wait few cycles to see the SUSPRDY = 1 acknowledge the suspend request.
  
  This bit is cleared by hardware simultaneous with the WAKEUP flag set.
  
  0: No effect
  
  1: Enter L1/L2 suspend

- **Condition: Host mode**
  
  Software can set this bit when host application has nothing scheduled for the next frames and wants to enter long term power saving. When set, it stops immediately SOF generation and any other host activity, gates the USB clock and sets the transceiver in low power mode. If any USB transaction is on-going at the time SUSPEN is set, suspend is entered at the end of the current transaction.
  
  As soon as suspend state is propagated internally and gets effective the SUSPRDY bit is set. In the case that host application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the micro-controller to STOP mode, it must first wait few cycles to see SUSPRDY=1 acknowledge to the suspend request.
  
  This bit is cleared by hardware simultaneous with the WAKEUP flag set.
  
  0: No effect
  
  1: Enter L1/L2 suspend

Bit 2 **SUSPRDY**: Suspend state effective

This bit is set by hardware as soon as the suspend state entered through the SUSPEN control gets internally effective. In this state USB activity is suspended, USB clock is gated, transceiver is set in low power mode by disabling the differential receiver. Only asynchronous wake-up logic and single ended receiver is kept alive to detect remote wake-up or resume events.

Software must poll this bit to confirm it to be set before any STOP mode entry.

This bit is cleared by hardware simultaneously to the WAKEUP flag being set.

  0: Normal operation
  
  1: Suspend state
Bit 1  **PDWN**: Power down

This bit is used to completely switch off all USB-related analog parts if it is required to completely disable the USB peripheral for any reason. When this bit is set, the USB peripheral is disconnected from the transceivers and it cannot be used.

0: Exit power down
1: Enter power down mode

Bit 0  **USBRST**: USB Reset

- Condition: Device mode

Software can set this bit to reset the USB core, exactly as it happens when receiving a RESET signaling on the USB. The USB peripheral, in response to a RESET, resets its internal protocol state machine. Reception and transmission are disabled until the RST_DCON bit is cleared. All configuration registers do not reset: the microcontroller must explicitly clear these registers (this is to ensure that the RST_DCON interrupt can be safely delivered, and any transaction immediately followed by a RESET can be completed). The function address and endpoint registers are reset by an USB reset event.

0: No effect
1: USB core is under reset

- Condition: Host mode

Software sets this bit to drive USB reset state on the bus and initialize the device. USB reset terminates as soon as this bit is cleared by software.

0: No effect
1: USB reset driven

**USB interrupt status register (USB_ISTR)**

Address offset: 0x44
Reset value: 0x0000 0000

This register contains the status of all the interrupt sources permitting application software to determine which events caused an interrupt request.

The upper part of this register contains single bits, each of them representing a specific event. These bits are set by the hardware when the related event occurs; if the corresponding bit in the USB_CNTR register is set, a generic interrupt request is generated. The interrupt routine, examining each bit, performs all necessary actions, and finally it clears the serviced bits. If any of them is not cleared, the interrupt is considered to be still pending, and the interrupt line is kept high again. If several bits are set simultaneously, only a single interrupt is generated.

Endpoint/channel transaction completion can be handled in a different way to reduce interrupt response latency. The CTR bit is set by the hardware as soon as an endpoint/channel successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB_CNTR is set. An endpoint/channel dedicated interrupt condition is activated independently from the CTRM bit in the USB_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB_CHEPnR register (the CTR bit is actually a read only bit). For endpoint-/channel-related interrupts, the software can use the direction of transaction (DIR) and IDN read-only bits to identify which endpoint/channel made the last interrupt request and called the corresponding interrupt service routine.

The user can choose the relative priority of simultaneously pending USB_ISTR events by specifying the order in which software checks USB_ISTR bits in an interrupt service routine. Only the bits related to events, which are serviced, are cleared. At the end of the service routine, another interrupt is requested, to service the remaining conditions.
To avoid spurious clearing of some bits, it is recommended to clear them with a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with 0 (these bits can only be cleared by software). Read-modify-write cycles must be avoided because between the read and the write operations another bit can be set by the hardware and the next write clears it before the device has the time to service the event.

| Bit 31 | Reserved, must be kept at reset value. |
| Bit 30 | **LS_DCON**: Low speed device connected |
| - Host mode: |
|          | This bit is set by hardware when an LS device connection is detected. Device connection is signaled after LS J-state is sampled for 22 consecutive cycles of the USB clock (48 MHz) from the unconnected state. |
| Bit 29 | **DCON_STAT**: Device connection status |
| - Host mode: |
|          | This bit contains information about device connection status. It is set by hardware when a LS/FS device is attached to the host while it is reset when the device is disconnected. |
|          | 0: No device connected |
|          | 1: FS or LS device connected to the host |
| Bit 28:18 | Reserved, must be kept at reset value. |
| Bit 17 | **DDISC**: Device connection |
| - Host mode: |
|          | This bit is set when a device connection is detected. This bit is read/write but only 0 can be written and writing 1 has no effect. |
| Bit 16 | **THR512**: 512 byte threshold interrupt |
|          | This bit is set to 1 by the hardware when 512 bytes have been transmitted or received during isochronous transfers. This bit is read/write but only 0 can be written and writing 1 has no effect. Note that no information is available to indicate the associated channel/endpoint, however in practice only one ISO endpoint/channel with such large packets can be supported, so that channel. |
| Bit 15 | **CTR**: Completed transfer in host mode |
|          | This bit is set by the hardware to indicate that an endpoint/channel has successfully completed a transaction; using DIR and IDN bits software can determine which endpoint/channel requested the interrupt. This bit is read-only. |
Bit 14 **PMAOV**: Packet memory area over / underrun
This bit is set if the microcontroller has not been able to respond in time to an USB memory request. The USB peripheral handles this event in the following way: During reception an ACK handshake packet is not sent, during transmission a bit-stuff error is forced on the transmitted stream; in both cases the host retries the transaction. The PMAOV interrupt must never occur during normal operations. Since the failed transaction is retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however this does not happen during isochronous transfers (no isochronous transaction is anyway retried) leading to a loss of data in this case. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 13 **ERR**: Error
This flag is set whenever one of the errors listed below has occurred:
- **NANS**: No ANSWer. The timeout for a host response has expired.
- **CRC**: Cyclic redundancy check error. One of the received CRCs, either in the token or in the data, was wrong.
- **BST**: Bit stuffing error. A bit stuffing error was detected anywhere in the PID, data, and/or CRC.
- **FVIO**: Framing format violation. A non-standard frame was received (EOP not in the right place, wrong token sequence, etc.).

The USB software can usually ignore errors, since the USB peripheral and the PC host manage retransmission in case of errors in a fully transparent way. This interrupt can be useful during the software development phase, or to monitor the quality of transmission over the USB bus, to flag possible problems to the user (for example loose connector, too noisy environment, broken conductor in the USB cable and so on). This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 12 **WKUP**: Wake-up
This bit is set to 1 by the hardware when, during suspend mode, activity is detected that wakes up the USB peripheral. This event asynchronously clears the SUSPRDY bit in the CTRLR register and activates the USB_WAKEUP line, which can be used to notify the rest of the device (for example wake-up unit) about the start of the resume process. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 11 **SUSP**: Suspend mode request
- **Device mode**
  This bit is set by the hardware when no traffic has been received for 3 ms, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset and it is disabled by the hardware when the suspend mode is active (SUSPEN=1) until the end of resume sequence. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 10 **RST_DCON**: USB reset request (Device mode) or device connect/disconnect (Host mode)
- **Device mode**
  This bit is set by hardware when an USB reset is released by the host and the bus returns to idle. USB reset state is internally detected after the sampling of 60 consecutive SE0 cycles.
- **Host mode**
  This bit is set by hardware when device connection or device disconnection is detected. Device connection is signaled after J state is sampled for 22 cycles consecutively from unconnected state. Device disconnection is signaled after SE0 state is seen for 22 bit times consecutively from connected state.
Bit 9 **SOF:** Start of frame
This bit signals the beginning of a new USB frame and it is set when a SOF packet arrives through the USB bus. The interrupt service routine may monitor the SOF events to have a 1 ms synchronization event to the USB host and to safely read the USB_FNR register which is updated at the SOF packet reception (this can be useful for isochronous applications). This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 8 **ESOF:** Expected start of frame
- **Device mode**
  This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each 1 ms, but if the device does not receive it properly, the suspend timer issues this interrupt. If three consecutive ESOF interrupts are generated (for example three SOF packets are lost) without any traffic occurring in between, a SUSP interrupt is generated. This bit is set even when the missing SOF packets occur while the suspend timer is not yet locked. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 7 **L1REQ:** LPM L1 state request
- **Device mode**
  This bit is set by the hardware when LPM command to enter the L1 state is successfully received and acknowledged. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bits 6:5 **Reserved,** must be kept at reset value.

Bit 4 **DIR:** Direction of transaction
This bit is written by the hardware according to the direction of the successful transaction, which generated the interrupt request.
- If DIR bit = 0, VTTX bit is set in the USB_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of IN type (data transmitted by the USB peripheral to the host PC).
- If DIR bit = 1, VTRX bit or both VTTX/VTRX are set in the USB_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of OUT type (data received by the USB peripheral from the host PC) or two pending transactions are waiting to be processed.
  This information can be used by the application software to access the USB_CHEPnR bits related to the triggering transaction since it represents the direction having the interrupt pending. This bit is read-only.

Bits 3:0 **IDN[3:0]:** Device Endpoint / host channel identification number
These bits are written by the hardware according to the host channel or device endpoint number, which generated the interrupt request. If several endpoint/channel transactions are pending, the hardware writes the identification number related to the endpoint/channel having the highest priority defined in the following way: two levels are defined, in order of priority: isochronous and double-buffered bulk channels/endpoints are considered first and then the others are examined. If more than one endpoint/channel from the same set is requesting an interrupt, the IDN bits in USB_ISTR register are assigned according to the lowest requesting register, CHEP0R having the highest priority followed by CHEP1R and so on. The application software can assign a register to each endpoint/channel according to this priority scheme, so as to order the concurring endpoint/channel requests in a suitable way. These bits are read only.
USB frame number register (USB_FNR)

Address offset: 0x48
Reset value: 0x0000 0XXX (where X is undefined)

### Bit Definitions

#### Bit 31:16
Reserved, must be kept at reset value.

- **Bit 15**: RXDP: Receive data + line status
  
  This bit can be used to observe the status of received data plus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

- **Bit 14**: RXDM: Receive data - line status
  
  This bit can be used to observe the status of received data minus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

- **Bit 13**: LCK: Locked
  
  - **Device mode**
    
    This bit is set by the hardware when at least two consecutive SOF packets have been received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer remains in this state until an USB reset or USB suspend event occurs.

- **Bits 12:11**: LSOF[1:0]: Lost SOF
  
  - **Device mode**
    
    These bits are written by the hardware when an ESOF interrupt is generated, counting the number of consecutive SOF packets lost. At the reception of an SOF packet, these bits are cleared.

- **Bits 10:0**: FN[10:0]: Frame number
  
  This bit field contains the 11-bits frame number contained in the last received SOF packet. The frame number is incremented for every frame sent by the host and it is useful for isochronous transfers. This bit field is updated on the generation of an SOF interrupt.

USB Device address (USB_DADDR)

Address offset: 0x4C
Reset value: 0x0000 0000
Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **EF**: Enable function

This bit is set by the software to enable the USB Device. The address of this device is contained in the following ADD[6:0] bits. If this bit is at 0 no transactions are handled, irrespective of the settings of USB_CHEPnR registers.

Bits 6:0 **ADD[6:0]**: Device address

- **Device mode**
  These bits contain the USB function address assigned by the host PC during the enumeration process. Both this field and the endpoint/channel address (EA) field in the associated USB_CHEPnR register must match with the information contained in a USB token in order to handle a transaction to the required endpoint.

- **Host mode**
  These bits contain the address transmitted with the LPM transaction

### LPM control and status register (USB_LPMCSR)

Address offset: 0x54

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **BESL[3:0]**: BESL value

- **Device mode**
  These bits contain the BESL value received with last ACKed LPM Token

Bit 3 **REMWAKE**: bRemoteWake value

- **Device mode**
  This bit contains the bRemoteWake value received with last ACKed LPM Token

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LPMACK**: LPM token acknowledge enable

- **Device mode**:
  0: the valid LPM token is NYET.
  1: the valid LPM token is ACK.

  The NYET/ACK is returned only on a successful LPM transaction:
  - No errors in both the EXT token and the LPM token (else ERROR)
  - A valid bLinkState = 0001B (L1) is received (else STALL)

Bit 0 **LPMEN**: LPM support enable

- **Device mode**
  This bit is set by the software to enable the LPM support within the USB Device. If this bit is at 0 no LPM transactions are handled.
Battery charging detector (USB_BCDR)

Address offset: 0x58
Reset value: 0x0000 0000

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **DPPU_DPD**: DP pull-up / DPDM pull-down
- **Device mode**
  This bit is set by software to enable the embedded pull-up on DP line. Clearing it to 0 can be used to signal disconnect to the host when needed by the user software.
- **Host mode**
  This bit is set by software to enable the embedded pull-down on DP and DM lines.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **PS2DET**: DM pull-up detection status
- **Device mode**
  This bit is active only during PD and gives the result of comparison between DM voltage level and $V_{LGC}$ threshold. In normal situation, the DM level must be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.
  0: Normal port detected (connected to SDP, ACA, CDP or DCP).
  1: PS2 port or proprietary charger detected.

Bit 6 **SDET**: Secondary detection (SD) status
- **Device mode**
  This bit gives the result of SD.
  0: CDP detected.
  1: DCP detected.

Bit 5 **PDET**: Primary detection (PD) status
- **Device mode**
  This bit gives the result of PD.
  0: no BCD support detected (connected to SDP or proprietary device).
  1: BCD support detected (connected to ACA, CDP or DCP).

Bit 4 **DCDET**: Data contact detection (DCD) status
- **Device mode**
  This bit gives the result of DCD.
  0: data lines contact not detected.
  1: data lines contact detected.
Host channel-specific/device endpoint-specific registers

The number of these registers varies according to the number of endpoints or host channels that the USB peripheral is designed to handle. The USB peripheral supports up to 8 bidirectional endpoints or host channels. Each USB Device must support a control endpoint/channel whose address (EA bits) must be set to 0. The USB peripheral behaves in an undefined way if multiple endpoints are enabled having the same endpoint/channel number value. For each endpoint, an USB_CHEPnR register is available to store the endpoint/channel specific information.

USB endpoint/channel n register (USB_CHEPnR)

Address offset: 0x00 + 0x4 * n, (n = 0 to 7)

Reset value: 0x0000 0000

They are also reset when an USB reset is received from the USB bus or forced through bit USBRST in the CTLR register, except the VTRX and VTTX bits, which are kept unchanged to avoid missing a correct packet notification immediately followed by an USB reset event. Each endpoint/channel has its USB_CHEPnR register where n is the endpoint/channel identifier.

Read-modify-write cycles on these registers must be avoided because between the read and the write operations some bits can be set by the hardware and the next write would modify them before the CPU has the time to detect the change. For this purpose, all bits affected by this problem have an ‘invariant’ value that must be used whenever their modification is not required. It is recommended to modify these registers with a load instruction where all the bits, which can be modified only by the hardware, are written with their ‘invariant’ value.
<table>
<thead>
<tr>
<th>Bits</th>
<th>Symbol</th>
<th>Description</th>
<th>Mode</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
<td>Reserved</td>
<td>Host mode</td>
<td>must be kept at reset value.</td>
</tr>
<tr>
<td>30-29</td>
<td>THREE_ERR_RX[1:0]</td>
<td>Three errors for an IN transaction</td>
<td>Host mode</td>
<td>This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an IN transaction. THREE_ERR_RX is not generated for isochronous transactions. The software can only clear this bit. Coding of the received error: 00: Less than 3 errors received. 01: More than 3 errors received, last error is timeout error. 10: More than 3 errors received, last error is data error (CRC error). 11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).</td>
</tr>
<tr>
<td>28-27</td>
<td>THREE_ERR_TX[1:0]</td>
<td>Three errors for an OUT or SETUP transaction</td>
<td>Host mode</td>
<td>This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an OUT transaction. THREE_ERR_TX is not generated for isochronous transactions. The software can only clear this bit. Coding of the received error: 00: Less than 3 errors received. 01: More than 3 errors received, last error is timeout error. 10: More than 3 errors received, last error is data error (CRC error). 11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).</td>
</tr>
<tr>
<td>26</td>
<td>ERR_RX</td>
<td>Received error for an IN transaction</td>
<td>Host mode</td>
<td>This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an IN transaction on this channel. The software can only clear this bit. If the ERRM bit in USB_CNTR register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.</td>
</tr>
<tr>
<td>25</td>
<td>ERR_TX</td>
<td>Received error for an OUT/SETUP transaction</td>
<td>Host mode</td>
<td>This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an OUT or SETUP transaction on this channel. The software can only clear this bit. If the ERRM bit in USB_CNTR register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.</td>
</tr>
</tbody>
</table>
Bit 24 **LS_EP**: Low speed endpoint – host with HUB only
- Host mode
  This bit is set by the software to send an LS transaction to the corresponding endpoint.
  0: Full speed endpoint
  1: Low speed endpoint

Bit 23 **NAK**:  
- Host mode
  This bit is set by the hardware when a device responds with a NAK. Software can use this bit to monitor the number of NAKs received from a device.

Bits 22:16 **DEVADDR[6:0]**:  
- Host mode
  Device address assigned to the endpoint during the enumeration process.

Bit 15 **VTRX**: USB valid transaction received  
- Device mode  
  This bit is set by the hardware when an OUT/SETUP transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated. The type of occurred transaction, OUT or SETUP, can be determined from the SETUP bit described below.  
  A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.
  This bit is read/write but only 0 can be written, writing 1 has no effect.  
- Host mode
  This bit is set by the hardware when an IN transaction is successfully completed on this channel. The software can only clear this bit. If the CTRM bit in USB_CNTR register is set a generic interrupt condition is generated together with the channel related flag, which is always activated.

- A transaction ended with a NAK sets this bit and NAK answer is reported to application reading the NAK state from the STATRX field of this register. One NAKed transaction keeps pending and is automatically retried by the host at the next frame, or the host can immediately retry by resetting STATRX state to VALID.

- A transaction ended by STALL handshake sets this bit and the STALL answer is reported to application reading the STALL state from the STATRX field of this register. Host application must consequently disable the channel and re-enumerate.

- A transaction ended with ACK handshake sets this bit
  If double buffering is disabled, ACK answer is reported by application reading the DISABLE state from the STATRX field of this register. Host application must read received data from USBRAM and re-arm the channel by writing VALID to the STATRX field of this register.
  If double buffering is enabled, ACK answer is reported by application reading VALID state from the STATRX field of this register. Host application must read received data from USBRAM and toggle the DTOGTX bit of this register.

- A transaction ended with error sets this bit.
  Errors can be seen via the bits ERR_RX (host mode only).

This bit is read/write but only 0 can be written, writing 1 has no effect.
Bit 14 **DTOGRX**: Data Toggle, for reception transfers

If the endpoint/channel is not isochronous, this bit contains the expected value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be received. Hardware toggles this bit, when the ACK handshake is sent following a data packet reception having a matching data PID value; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP PID received from host (in device mode), while it sets this bit to 1 when SETUP transaction is acknowledged by device (in host mode).

If the endpoint/channel is using the double-buffering feature this bit is used to support packet buffer swapping too (Refer to Section 55.5.3: Double-buffered endpoints and usage in Device mode).

If the endpoint/channel is isochronous, this bit is used only to support packet buffer swapping for data transmission since no data toggling is used for this kind of channels/endpoints and only DATA0 packet are transmitted (Refer to Section 55.5.5: Isochronous transfers in Device mode). Hardware toggles this bit just after the end of data packet reception, since no handshake is used for isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGRX remains unchanged, while writing 1 makes the bit value toggle. This bit is read/write but it can be only toggled by writing 1.
Bits 13:12 **STATRX[1:0]:** Status bits, for reception transfers

- **Device mode**
  
  These bits contain information about the endpoint status, which are listed in **Table 598: Reception status encoding on page 2618**. These bits can be toggled by software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATRX bits to NAK when a correct transfer has occurred (VTRX = 1) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledges a new transaction.

  Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to **Section 55.5.3: Double-buffered endpoints and usage in Device mode**).

  If the endpoint is defined as isochronous, its status can be only "VALID" or "DISABLED", so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STATRX bits to ‘STALL’ or ‘NAK’ for an isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

- **Host mode**
  
  These bits are the host application controls to start, retry, or abort host transactions driven by the channel.

  These bits also contain information about the device answer to the last IN channel transaction and report the current status of the channel according to the following STATRX table of states:

  - **DISABLE**
    
    DISABLE value is reported in case of ACK acknowledge is received on a single-buffer channel. When in DISABLE state the channel is unused or not active waiting for application to restart it by writing VALID. Application can reset a VALID channel to DISABLE to abort a transaction. In this case the transaction is immediately removed from the host execution list. If the aborted transaction was already under execution it is regularly terminated on the USB but the relative VTRX interrupt is not generated.

  - **VALID**
    
    A host channel is actively trying to submit USB transaction to device only when in VALID state. VALID state can be set by software or automatically by hardware on a NAKED channel at the start of a new frame. When set to VALID, an host channel enters the host execution queue and waits permission from the host frame scheduler to submit its configured transaction.

    VALID value is also reported in case of ACK acknowledge is received on a double-buffered channel. In this case the channel remains active on the alternate buffer while application needs to read the current buffer and toggle DTOGTX. In case software is late in reading and the alternate buffer is not ready, the host channel is automatically suspended transparently to the application. The suspended double buffered channel is re-activated as soon as delay is recovered and DTOGTX is toggled.

  - **NAK**
    
    NAK value is reported in case of NAK acknowledge received. When in NAK state the channel is suspended and does not try to transmit. NAK state is moved to VALID by hardware at the start of the next frame, or software can change it to immediately retry transmission by writing it to VALID, or can disable it and abort the transaction by writing DISABLE.

  - **STALL**
    
    STALL value is reported in case of STALL acknowledge received. When in STALL state the channel behaves as disabled. Application must not retry transmission but reset the USB and re-enumerate.
Bit 11 **SETUP**: Setup transaction completed
  
  **– Device mode**
  
  This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (VTRX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while VTRX bit is at 1; its state changes when VTRX is at 0. This bit is read-only.
  
  **– Host mode**
  
  This bit is set by the software to send a SETUP transaction on a control endpoint. This bit changes its value only for control endpoints. It is cleared by hardware when the SETUP transaction is acknowledged and VTTX interrupt generated.

Bits 10:9 **UTYPE[1:0]**: USB type of transaction

These bits configure the behavior of this endpoint/channel as described in *Table 599: Endpoint/channel type encoding*. Channel0/Endpoint0 must always be a control endpoint/channel and each USB function must have at least one control endpoint/channel which has address 0, but there may be other control channels/endpoints if required. Only control channels/endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint/channel is defined as NAK, the USB peripheral does not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint/channel is defined as STALL in the receive direction, then the SETUP packet is accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint/channel is a control one. Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EPKIND configuration bit.

The usage of isochronous channels/endpoints is explained in *Section 55.5.5: Isochronous transfers in Device mode*

Bit 8 **EPKIND**: endpoint/channel kind

The meaning of this bit depends on the endpoint/channel type configured by the UTYPE bits. *Table 600* summarizes the different meanings.

**DBL_BUF**: This bit is set by the software to enable the double-buffering feature for this bulk endpoint. The usage of double-buffered bulk endpoints is explained in *Section 55.5.3: Double-buffered endpoints and usage in Device mode*.

**STATUS_OUT**: This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered ‘STALL’ instead of ‘ACK’. This bit may be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS_OUT is reset, OUT transactions can have any number of bytes, as required.

Bit 7 **VTTX**: Valid USB transaction transmitted

  **– Device mode**
  
  This bit is set by the hardware when an IN transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in the USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated.

  A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

  This bit is read/write but only 0 can be written.
  
  **– Host mode**
  
  Same as VTRX behavior but for USB OUT and SETUP transactions.
Bit 6 **DTOGTX**: Data toggle, for transmission transfers

If the endpoint/channel is non-isochronous, this bit contains the required value of the data toggle bit \((0 = \text{DATA0}, 1 = \text{DATA1})\) for the next data packet to be transmitted. Hardware toggles this bit when the ACK handshake is received from the USB host, following a data packet transmission. If the endpoint/channel is defined as a control one, hardware sets this bit to 1 at the reception of a SETUP PID addressed to this endpoint (in device mode) or when a SETUP transaction is acknowledged by the device (in host mode).

If the endpoint/channel is using the double buffer feature, this bit is used to support packet buffer swapping too (Refer to Section 55.5.3: Double-buffered endpoints and usage in Device mode).

If the endpoint/channel is isochronous, this bit is used to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (refer to Section 55.5.5: Isochronous transfers in Device mode). Hardware toggles this bit just after the end of data packet transmission, since no handshake is used for isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint/channel is not a control one) or to force a specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGTX remains unchanged, while writing 1 makes the bit value to toggle. This bit is read/write but it can only be toggled by writing 1.

Bits 5:4 **STATTX\[1:0\]**: Status bits, for transmission transfers

- **Device mode**
  
  These bits contain the information about the endpoint status, listed in Table 601. These bits can be toggled by the software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATTX bits to NAK, when a correct transfer has occurred \((VTTX = 1)\) corresponding to a IN or SETUP (control only) transaction addressed to this channel/endpoint. It then waits for the software to prepare the next set of data to be transmitted.

  Double-buffered bulk endpoints implement a special transaction flow control, which controls the status based on buffer availability condition (Refer to Section 55.5.3: Double-buffered endpoints and usage in Device mode).

  If the endpoint is defined as isochronous, its status can only be "VALID" or "DISABLED". Therefore, the hardware cannot change the status of the channel/endpoint/channel after a successful transaction. If the software sets the STATTX bits to 'STALL' or 'NAK' for an isochronous channel/endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can only be toggled by writing 1.

- **Host mode**

  The STATTX bits contain the information about the channel status. Refer to Table 601 for the full descriptions ("Host mode" descriptions). Whereas in Device mode, these bits contain the status that are given out on the following transaction, in Host mode they capture the status last received from the device. If a NAK is received, STATTX contains the value indicating NAK.

Bits 3:0 **EA\[3:0\]**: endpoint/channel address

- **Device mode**

  Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. A value must be written before enabling the corresponding endpoint.

- **Host mode**

  Software must write in this field the 4-bit address used to identify the channel addressed by the host transaction.
### Table 598. Reception status encoding

<table>
<thead>
<tr>
<th>STATRX[1:0]</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td><strong>DISABLED</strong>: all reception requests addressed to this endpoint/channel are ignored.</td>
</tr>
<tr>
<td>01</td>
<td><strong>STALL</strong>: Device mode: the endpoint is stalled and all reception requests result in a STALL handshake. Host mode: this indicates that the device has STALLed the channel.</td>
</tr>
<tr>
<td>10</td>
<td><strong>NAK</strong>: Device mode: the endpoint is NAKed and all reception requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the reception request.</td>
</tr>
<tr>
<td>11</td>
<td><strong>VALID</strong>: this endpoint/channel is enabled for reception.</td>
</tr>
</tbody>
</table>

### Table 599. Endpoint/channel type encoding

<table>
<thead>
<tr>
<th>UTYP[$1:0$]</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>BULK</td>
</tr>
<tr>
<td>01</td>
<td>CONTROL</td>
</tr>
<tr>
<td>10</td>
<td>ISO</td>
</tr>
<tr>
<td>11</td>
<td>INTERRUPT</td>
</tr>
</tbody>
</table>

### Table 600. Endpoint/channel kind meaning

<table>
<thead>
<tr>
<th>UTYP[$1:0$]</th>
<th>EPKIND meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>BULK DBL_BUF</td>
</tr>
<tr>
<td>01</td>
<td>CONTROL STATUS_OUT</td>
</tr>
<tr>
<td>10</td>
<td>ISO SBUF_ISO: This bit is set by the software to enable the single-buffering feature for isochronous endpoint</td>
</tr>
<tr>
<td>11</td>
<td>INTERRUPT Not used</td>
</tr>
</tbody>
</table>

### Table 601. Transmission status encoding

<table>
<thead>
<tr>
<th>STATTX[1:0]</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td><strong>DISABLED</strong>: all transmission requests addressed to this endpoint/channel are ignored.</td>
</tr>
<tr>
<td>01</td>
<td><strong>STALL</strong>: Device mode: the endpoint is stalled and all transmission requests result in a STALL handshake. Host mode: this indicates that the device has STALLed the channel.</td>
</tr>
</tbody>
</table>
Device mode: the endpoint is NAKed and all transmission requests result in a NAK handshake.
Host mode: this indicates that the device has NAKed the transmission request.

### Table 601. Transmission status encoding (continued)

<table>
<thead>
<tr>
<th>STATTX[1:0]</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td><strong>NAK</strong>: Device mode: the endpoint is NAKed and all transmission requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the transmission request.</td>
</tr>
<tr>
<td>11</td>
<td><strong>VALID</strong>: this endpoint/channel is enabled for transmission.</td>
</tr>
</tbody>
</table>
55.6.2 Buffer descriptor table

**Note:** The buffer descriptor table is located inside the packet buffer memory in the separate "USB SRAM" address space.

Although the buffer descriptor table is located inside the packet buffer memory ("USB SRAM" area), its entries can be considered as additional registers used to configure the location and size of the packet buffers used to exchange data between the USB macro cell and the device.

The first packet memory location is located at USB SRAM base address. The buffer descriptor table entry associated with the USB_CHEPnR registers is described below. The memory must be addressed using Word (32-bit) accesses.

A thorough explanation of packet buffers and the buffer descriptor table usage can be found in *Structure and usage of packet buffers on page 2586*.

**Channel/endpoint transmit buffer descriptor n (USB_CHEP_TXRXBD_n)**

Address offset: n*8

This register description applies when corresponding CHEPnR register does not program the use of double buffering working in receive mode (otherwise refer to following register description)

| Bits 31:26 Reserved, must be kept at reset value. |
| Bits 25:16 **COUNT_TX[9:0]**: Transmission byte count |
| These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. |
| Bits 15:0 **ADDR_TX[15:0]**: Transmission buffer address |
| These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as "00" since packet memory is word wide and all packet buffers must be word aligned. |

**Channel/endpoint receive buffer descriptor n [alternate] (USB_CHEP_TXRXBD_n)**

Address offset: n*8

This register description applies when corresponding CHEPnR register programs the use of double buffering and activates receive buffers (otherwise refer to previous register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to
the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

<table>
<thead>
<tr>
<th>BLSIZE</th>
<th>NUM_BLOCK[4:0]</th>
<th>COUNT_RX[9:0]</th>
<th>ADDR_RX[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31  **BLSIZE**: Block size

This bit selects the size of memory block used to define the allocated buffer area.

- If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.
- If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26  **NUM_BLOCK[4:0]**: Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in Table 602.

Bits 25:16  **COUNT_RX[9:0]**: Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB_CHEPnR register during the last OUT/SETUP transaction addressed to it.

Bits 15:0  **ADDR_RX[15:0]**: Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

**Table 602. Definition of allocated buffer memory**

<table>
<thead>
<tr>
<th>Value of NUM_BLOCK[4:0]</th>
<th>Memory allocated when BLSIZE=0</th>
<th>Memory allocated when BLSIZE=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (00000)</td>
<td>Not allowed</td>
<td>32 bytes</td>
</tr>
<tr>
<td>1 (00001)</td>
<td>2 bytes</td>
<td>64 bytes</td>
</tr>
<tr>
<td>2 (00010)</td>
<td>4 bytes</td>
<td>96 bytes</td>
</tr>
<tr>
<td>3 (00011)</td>
<td>6 bytes</td>
<td>128 bytes</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>14 (01110)</td>
<td>28 bytes</td>
<td>480 bytes</td>
</tr>
<tr>
<td>15 (01111)</td>
<td>30 bytes</td>
<td></td>
</tr>
</tbody>
</table>
Table 602. Definition of allocated buffer memory (continued)

<table>
<thead>
<tr>
<th>Value of NUM_BLOCK[4:0]</th>
<th>Memory allocated when BLSIZE=0</th>
<th>Memory allocated when BLSIZE=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 (10000)</td>
<td>32 bytes</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>29 (11101)</td>
<td>58 bytes</td>
<td></td>
</tr>
<tr>
<td>30 (11110)</td>
<td>60 bytes</td>
<td>992 bytes</td>
</tr>
<tr>
<td>31 (11111)</td>
<td>62 bytes</td>
<td>1023 bytes</td>
</tr>
</tbody>
</table>

Channel/endpoint receive buffer descriptor n
(USB_CHEP_RTXXBD_n)

Address offset: n*8 + 4

This register description applies when corresponding CHEPnR register does not program use of double buffering in the transmit mode (otherwise refer to following register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).
Bit 31  **BLSIZE**: Block size

This bit selects the size of memory block used to define the allocated buffer area.

- If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.
- If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26 **NUM_BLOCK[4:0]**: Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in Table 602.

Bits 25:16 **COUNT_RX[9:0]**: Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB_CHEPnR register during the last OUT/SETUP transaction addressed to it.

Bits 15:0 **ADDR_RX[15:0]**: Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as "00" since packet memory is word wide and all packet buffers must be word aligned.

**Channel/endpoint transmit buffer descriptor n [alternate]**

(USB_CHEP_RXTXBD_n)

Address offset: n*8 + 4

This register description applies when corresponding CHEPnR register programs use of double buffering and activates transmit buffers (otherwise refer to previous register description).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COUNT_TX[9:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **COUNT_TX[9:0]**: Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR_TX[15:0]**: Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as "00" since packet memory is word wide and all packet buffers must be word aligned.
## 55.6.3 USB register map

The table below provides the USB register map and reset values.

| Offset | Register  | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|-----------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x00   | USB_CHEP0R|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x04   | USB_CHEP1R|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x08   | USB_CHEP2R|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x0C   | USB_CHEP3R|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x10   | USB_CHEP4R|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x14   | USB_CHEP5R|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x18   | USB_CHEP6R|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1C   | USB_CHEP7R|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

Table 603. USB register map and reset values

The table above provides the USB register map and reset values.
Table 603. USB register map and reset values (continued)

| Offset | Register     | Host | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|--------------|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x20-  |              |      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x40   | USB_CNTR     | 0x00|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |              |     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x44   | USBISTR      | 0x00|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x48   | USB_FNR      | 0x00|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4C   | USB_DADDR    | 0x00|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x54   | USB_LMCSR    | 0x00|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x58   | USB_BCDR     | 0x00|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.3 on page 111 for the register boundary addresses.
56 USB Type-C®/USB Power Delivery interface (UCPD)

56.1 Introduction

The USB Type-C/USB Power Delivery interface complies with:
- Universal Serial Bus Type-C Cable and Connector Specification: release 2.2, Oct 2022
- Universal Serial Bus Power Delivery specifications:
  - revision 2.0, version 1.3, January 12, 2017
  - revision 3.1, version 1.6, October 2022

It integrates the physical layer of the Power Delivery (PD) specification, with CC signaling method (no VBUS), for operation with Type-C cables.

56.2 UCPD main features

- Compliance with USB Type-C specification release 2.2
- Compliance with USB Power Delivery specifications revision 2.0 and 3.1
  - Enabling advanced applications such as PPS (programmable power supply)
- Stop mode low-power operation support
- Built-in analog PHY
  - USB Type-C pull-up (Rp, all values) and pull-down (Rd) resistors
  - “Dead battery” Rd support
  - USB Power Delivery message transmission and reception
  - FRS (fast role swap) Rx support
- Digital controller
  - BMC (bi-phase mark coding) encode and decode
  - 4b5b encode and decode
  - USB Type-C level detection with de-bounce, generating interrupts
  - FRS signaling
  - FRS detection, generating an interrupt
  - DMA-compatible byte-level interface for USB Power Delivery payload, generating interrupts
  - USB Power Delivery clock pre-scaler / dividers
  - CRC generation/checking
  - Support of ordered sets, with a programmable ordered set mask at receive
  - Clock recovery from incoming Rx stream

56.3 UCPD implementation

The devices have one UCPD controller to support one USB Type-C port.
The following table gives the memory locations of trim data stored in the non-volatile memory, to use in the software trimming procedure described in Section 56.5.5: UCPD software trimming.

Table 605. UCPD software trim data

<table>
<thead>
<tr>
<th>Name</th>
<th>Non-volatile memory location</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Address</td>
</tr>
<tr>
<td>3A0_CC1[3:0]</td>
<td>0x4002 242C</td>
</tr>
<tr>
<td>3A0_CC2[3:0]</td>
<td>0x4002 242C</td>
</tr>
<tr>
<td>1A5_CC1[3:0]</td>
<td>0x08FF F844</td>
</tr>
<tr>
<td>1A5_CC2[3:0]</td>
<td>0x08FF F844</td>
</tr>
<tr>
<td>Rd_CC1[3:0]</td>
<td>0x4002 242C</td>
</tr>
<tr>
<td>Rd_CC2[3:0]</td>
<td>0x4002 242C</td>
</tr>
</tbody>
</table>

56.4 UCPD functional description

The UCPD peripheral provides hardware support for the USB Power Delivery control interface specification, using I/Os specifically designed for that purpose.

The built-in PHY directly detects Type-C voltage levels, supports Power Delivery BIST carrier mode 2 (Tx only), BIST test data (Tx and Rx), and Power Delivery Rx FRS signaling.

For Power Delivery FRS Tx signaling, the device can be configured to control, through UCPD_FRSTX pin (alternate function), external NMOS transistors that ensure low-resistance pull-down on CC lines.

The UCPD transmitter BMC (bi-phase mark) encodes and transmits data: preamble, SOP, payload data from protocol layer (after 4b5b-encoding), CRC, and EOP on the Type-C connector CC lines. It automatically inserts inter-frame gap and executes “Hard Reset”.

The UCPD receiver detects SOP, BMC-decodes the incoming stream, recovers the preamble, 4b5b-decodes payload data, detects EOP, and checks CRC. It automatically detects five K-code SOP and two Reset ordered sets, plus two software-defined patterns (allows for only three out of four K-codes being correctly received, as defined by the standard).
In Stop mode, the peripheral maintains the ability to detect incoming USB Power Delivery messages and FRS signaling, which allows low-power operation.

### 56.4.1 UCPD block diagram

#### Figure 782. UCPD block diagram

The following table lists external signals (alternate or additional I/O functions).

#### Table 606. UCPD signals on pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCPDx_FRSTX</td>
<td>Output</td>
<td>USB Type-C fast role swap (FRS) signaling control, applicable to DRPs only. The signal (active high) drives an external NMOS transistor that pulls down the active CC line. A typical application has two such transistors (one per CC line) and reserves a separate I/O to drive either NMOS. Initially, the I/Os are configured as low-driving GPIOs. Upon detecting, through the Type-C state machine, the orientation of the cable attached, which determines the active CC line, the I/O of the active CC line must be set to its UCPDx_FRSTX alternate function and the I/O of the inactive CC line as low-driving GPIO.</td>
</tr>
<tr>
<td>UCPDx_CC1</td>
<td>Input/output</td>
<td>USB Type-C configuration control line 1, to be routed to the USB Type-C connector CC1 terminal.</td>
</tr>
<tr>
<td>UCPDx_CC2</td>
<td>Input/output</td>
<td>USB Type-C configuration control line 2, to be routed to the USB Type-C connector CC2 terminal.</td>
</tr>
</tbody>
</table>
The following table lists key internal signals.

**Table 606. UCPD internal signals**

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucpd_pclk</td>
<td>Input</td>
<td>APB clock for registers</td>
</tr>
<tr>
<td>ucpd_ker_ck</td>
<td>Input</td>
<td>Kernel clock</td>
</tr>
<tr>
<td>ucpd_tx_dma</td>
<td>Input/Output</td>
<td>Rx DMA acknowledge / request</td>
</tr>
<tr>
<td>ucpd_rx_dma</td>
<td>Input/Output</td>
<td>Tx DMA acknowledge / request</td>
</tr>
<tr>
<td>ucpd_it</td>
<td>Output</td>
<td>Interrupt request (all interrupts OR-ed) connected to NVIC</td>
</tr>
<tr>
<td>ucpd_wkup</td>
<td>Output</td>
<td>Wake-up request connected to EXTI</td>
</tr>
<tr>
<td>clk_rq</td>
<td>Output</td>
<td>Clock request connected to RCC</td>
</tr>
</tbody>
</table>

### 56.4.2 UCPD reset and clocks

The peripheral has a single reset signal (APB bus reset).

The register section is clocked with the APB clock (ucpd_pclk).

The main functional part of the transmitter is clocked with ucpd_clk clock, pre-scaled from the ucpd_ker_ck (HSI16) clock according to the PSC_USBPDCLK[2:0] bitfield of the UCPD_CFGR1 register. The main functional part of the receiver is clocked with the ucpd_rx_clk recovered from the incoming bitstream.

The receiver is designed to work in the clock frequency range from 6 to 18 MHz. However, the optimum performance is ensured in the range from 9 to 18 MHz.

The following diagram shows the clocking and timing elements of the UCPD peripheral.
Refer to the USB PD specification in order to set appropriate delays. For \( t_{\text{TransitionWindow}} \) and especially for \( t_{\text{InterFrameGap}} \), the clock frequency uncertainty must be taken into account so as to respect specified timings in all cases.

### 56.4.3 Physical layer protocol

The physical layer covers the signaling underlying the USB Power Delivery specification.

On the transmitter side its main function is to form packets according to the defined packet format including generally:
- preamble
- start of packet (SOP, ordered set)
- payload header
- payload data
- cyclic redundancy check (CRC) information
- end of packet (EOP)

Before going on the CC line, the data stream is BMC-encoded, respecting specified timing restrictions.

On the receive side, the principle task is to:
- extract start of packet (SOP, ordered set) information
- extract payload header
- extract payload data
- receive and check CRC
- receive end of packet (EOP)

The receive is basically a reverse of the transmit process, thus starting with BMC data stream decoding.

**Symbol encoding**

Apart from the preamble all symbols are encoded with a 4b5b scheme according to the specification shown in the following table.
## Table 608. 4b5b symbol encoding table

<table>
<thead>
<tr>
<th>Name</th>
<th>4b</th>
<th>5b</th>
<th>Symbol description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>1110</td>
<td>hex data 0</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>0100</td>
<td>hex data 1</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>1010</td>
<td>hex data 2</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>1010</td>
<td>hex data 3</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>0100</td>
<td>hex data 4</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>0100</td>
<td>hex data 5</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>1110</td>
<td>hex data 6</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>1110</td>
<td>hex data 7</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1110</td>
<td>hex data 8</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>0111</td>
<td>hex data 9</td>
</tr>
<tr>
<td>A</td>
<td>1010</td>
<td>1010</td>
<td>hex data A</td>
</tr>
<tr>
<td>B</td>
<td>1011</td>
<td>1011</td>
<td>hex data B</td>
</tr>
<tr>
<td>C</td>
<td>1100</td>
<td>1100</td>
<td>hex data C</td>
</tr>
<tr>
<td>D</td>
<td>1101</td>
<td>0011</td>
<td>hex data D</td>
</tr>
<tr>
<td>E</td>
<td>1110</td>
<td>0001</td>
<td>hex data E</td>
</tr>
<tr>
<td>F</td>
<td>1111</td>
<td>1111</td>
<td>hex data F</td>
</tr>
<tr>
<td>Sync-1</td>
<td>K-code</td>
<td>1100</td>
<td>Startsynch #1</td>
</tr>
<tr>
<td>Sync-2</td>
<td>K-code</td>
<td>1000</td>
<td>Startsynch #2</td>
</tr>
<tr>
<td>RST-1</td>
<td>K-code</td>
<td>0011</td>
<td>Hard Reset #1</td>
</tr>
<tr>
<td>RST-2</td>
<td>K-code</td>
<td>1100</td>
<td>Hard Reset #2</td>
</tr>
<tr>
<td>EOP</td>
<td>K-code</td>
<td>0110</td>
<td>EOP</td>
</tr>
<tr>
<td>Reserved</td>
<td>Error</td>
<td>0000</td>
<td>Do Not Use</td>
</tr>
<tr>
<td>Reserved</td>
<td>Error</td>
<td>0001</td>
<td>Do Not Use</td>
</tr>
<tr>
<td>Reserved</td>
<td>Error</td>
<td>0010</td>
<td>Do Not Use</td>
</tr>
<tr>
<td>Reserved</td>
<td>Error</td>
<td>0011</td>
<td>Do Not Use</td>
</tr>
<tr>
<td>Reserved</td>
<td>Error</td>
<td>0010</td>
<td>Do Not Use</td>
</tr>
<tr>
<td>Reserved</td>
<td>Error</td>
<td>0110</td>
<td>Do Not Use</td>
</tr>
<tr>
<td>Reserved</td>
<td>Error</td>
<td>0100</td>
<td>Do Not Use</td>
</tr>
<tr>
<td>Reserved</td>
<td>Error</td>
<td>1111</td>
<td>Do Not Use</td>
</tr>
</tbody>
</table>
Ordered sets

An ordered set consists of four K-codes as shown in the following figure.

Figure 784. K-code transmission

The following table lists the defined ordered sets, including all possible SOP* sequences. At the physical layer, the Hard Reset has higher priority than the other ordered sets so it can interrupt an ongoing Tx message.

Table 609. Ordered sets

<table>
<thead>
<tr>
<th>Ordered set name</th>
<th>K-code #1</th>
<th>K-code #2</th>
<th>K-code #3</th>
<th>K-code #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td>Sync-1</td>
<td>Sync-1</td>
<td>Sync-1</td>
<td>Sync-2</td>
</tr>
<tr>
<td>SOP'</td>
<td>Sync-1</td>
<td>Sync-1</td>
<td>Sync-3</td>
<td>Sync-3</td>
</tr>
<tr>
<td>SOP''</td>
<td>Sync-1</td>
<td>Sync-3</td>
<td>Sync-1</td>
<td>Sync-3</td>
</tr>
<tr>
<td>Hard Reset</td>
<td>RST-1</td>
<td>RST-1</td>
<td>RST-1</td>
<td>RST-2</td>
</tr>
<tr>
<td>Cable Reset</td>
<td>RST-1</td>
<td>Sync-1</td>
<td>RST-1</td>
<td>Sync-3</td>
</tr>
<tr>
<td>SOP' _Debug</td>
<td>Sync-1</td>
<td>RST-2</td>
<td>RST-2</td>
<td>Sync-3</td>
</tr>
<tr>
<td>SOP'' _Debug</td>
<td>Sync-1</td>
<td>RST-2</td>
<td>Sync-3</td>
<td>Sync-2</td>
</tr>
</tbody>
</table>

On reception, the physical layer must accept ordered sets with any combination of three correct K-codes out of four, as shown in the following table:

Table 610. Validation of ordered sets

<table>
<thead>
<tr>
<th>Status</th>
<th>1st code</th>
<th>2nd code</th>
<th>3rd code</th>
<th>4th code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>Corrupt</td>
<td>K-code</td>
<td>K-code</td>
<td>K-code</td>
</tr>
<tr>
<td>Valid</td>
<td>K-code</td>
<td>Corrupt</td>
<td>K-code</td>
<td>K-code</td>
</tr>
</tbody>
</table>
Bit ordering at transmission

Allowed transmission data units / data sizes are in the following table.

### Table 611. Data size

<table>
<thead>
<tr>
<th>Data unit</th>
<th>Non-encoded</th>
<th>Encoded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>8-bits</td>
<td>10-bits</td>
</tr>
<tr>
<td>Word</td>
<td>16-bits</td>
<td>20-bits</td>
</tr>
<tr>
<td>DWord</td>
<td>32-bits</td>
<td>40-bits</td>
</tr>
</tbody>
</table>

The bit transmission order is shown in the following figure.

**Figure 785. Transmit order for various sizes of data**
Packet format

Messages other than Hard Reset and Cable Reset

The packet format is shown in the following figure, with information on 4b5b encode and data source.

**Figure 786. Packet format**

<table>
<thead>
<tr>
<th>Preamble (training for receiver)</th>
<th>SOP* (start of packet)</th>
<th>Header</th>
<th>Byte 0</th>
<th>Byte 1</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td>Byte n-1</td>
<td>Byte n</td>
<td>CRC</td>
<td>EOP (end of packet)</td>
</tr>
</tbody>
</table>

Legend:
- Provided by the physical layer, not 4b5b-encoded
- Provided by the physical layer, 4b5b-encoded
- Provided by the protocol layer, 4b5b-encoded

**Hard Reset**

The physical layer handles the Hard Reset signaling differently than the other types of message as it has higher priority to be able to interrupt an ongoing transfer.

The physical layer specification implies the following sequence in the case of an ongoing Tx message:

1. Terminate the message by sending an EOP K-code and discard the rest of the message.
2. Wait for $t_{InterFrameGap}$ time.
3. If the CC line is not idle, wait until it goes idle.
4. Send the preamble followed by the four K-codes of Hard Reset signaling.
5. Disable the CC channel (stop sending and receiving), reset the physical layer and inform the protocol layer that the physical layer is reset.
6. Re-enable the channel when requested by the protocol layer.

**Figure 787. Line format of Hard Reset**

<table>
<thead>
<tr>
<th>Preamble (training for receiver)</th>
<th>RST-1</th>
<th>RST-1</th>
<th>RST-1</th>
<th>RST-2</th>
</tr>
</thead>
</table>

Legend:
- Provided by the physical layer, not 4b5b-encoded
- Provided by the physical layer, 4b5b-encoded
Cable Reset

Cable Reset shown in the following figure is similar in format to Hard Reset, but unlike Hard Reset it does not require a specific high-priority treatment.

**Figure 788. Line format of Cable Reset**

<table>
<thead>
<tr>
<th>Preamble</th>
<th>RST-1</th>
<th>Sync-1</th>
<th>RST-1</th>
<th>Sync-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(training for receiver)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cable Reset ordered set</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- Provided by the physical layer, not 4b5b-encoded
- Provided by the physical layer, 4b5b-encoded

Collision avoidance

The physical layer respects the $t_{InterFrameGap}$ delay between end of last-transmitted bit of a Tx message, and the first bit of a following message.

It also checks the idle state of the CC line before starting transmission. The CC line is considered idle if it shows less than three ($n_{TransitionCount}$) transitions within $t_{TransitionWindow}$ (12 to 20 μs). The Power Delivery specification revision 3.1 also requires to manage the $R_p$ value (source) and monitor Type-C voltage level for these $R_p$ modifications (at the sink).

Physical layer signaling schemes

The bit are signaled with bi-phase mark coding (BMC).

BIST

Depending on the BIST action required by the protocol layer, either of the following can be run:
- a Tx BIST pattern test, achieved by writing TXMODE and TXSEND
- an Rx BIST pattern test, achieved by writing RXMODE to the correct value for RXBIST.

The two possible patterns supported in UCPD (corresponding to "BMC" mode) are:
- BIST Test Data (192 bit pattern), applies to Tx and Rx. In the case of Rx, the message is received (but discarded rather than passing to the protocol layer, which must nevertheless still generate a GoodCRC Tx message in acknowledgment).
- BIST Carrier Mode 2 (single pattern, infinite length message), applies to Tx only. As opposed to Tx, the receiver in this mode simply ignores the CC line during this state.

BIST test data pattern

The test data pattern is not viewed as a special case in UCPD.
The BIST test data packet frame format is shown in the following figure.

**Figure 789. BIST test data frame**

![BIST test data frame diagram](image)

This is a fixed length test data pattern. In reality the only aspect that marks its difference from the general packet format already shown in *Figure 786: Packet format* is the contents of the Header. As UCPD receives the Tx Header contents via programming (it is simply viewed as part of the payload), it is only this programming (and not the block’s behavior) that differentiates the general packet from the BIST Test Data packet.

**BIST Carrier Mode 2**

When required, this BIST test mode sends an alternating pattern of 1010 that is continually repeated. As this mode is intended for signal analysis it is stable condition with (in V1.0 of the USB PD specification) no defined length. Starting from V1.1 of the USB PD specification, the protocol layer defines a counter that indicates when to exit this mode.

The way to quit the infinite 1010 sequence (according to requirements of the USB PD specification) is to disable the UCPD peripheral via the UCPDEN bit.

**Figure 790. BIST Carrier Mode 2 frame**

![BIST Carrier Mode 2 frame diagram](image)

### 56.4.4 UCPD BMC transmitter

The BMC transmitter comprises 4b5b encoding, CRC generation, and BMC encode, as shown in the following figure. Its output goes to the analog PHY through a channel switch.
BMC encoder


The half-bit clock hbit_clk is derived from ucpd_clk through a simple divider controlled by the HBITCLKDIV[5:0] bitfield of the UCPD_CFGR1 register. This ensures the same duration of high and low half-bit periods (if neglecting a small difference due to different rising and falling edge duration and due to jitter), and the same bit duration (if neglecting jitter).

Transmitter timing and collision avoidance

Hardware support of collision avoidance is made as a function of the half bit time for the transmitter. Two counters are implemented:

- $t_{\text{InterFrameGap}}$: via IFRGAP (pre-defined value, can be altered)
- $t_{\text{TransitionWindow}}$: via TRANSWIN (pre-defined value, can be altered)

These two counters once set correctly generates the interframe gap.

Hard Reset in transmitter

In order to facilitate generation of a Hard Reset, a special code of TXMODE field is used. No other fields need to be written.

On writing the correct code, the hardware forces Hard Reset Tx under the correct (optimal) timings with respect to an ongoing Tx message, which (if still in progress) is cleanly terminated by truncating the current Tx message and directly appending an EOP K-code sequence. No specific interrupt is generated relating to this truncation event.

Transmitter behavior in the case of errors

The under-run condition (TXUND interrupt) may happen by accident and in this case, the UCPD is starved of (the correct) Tx payload and is not able to complete the Tx message correctly. This is a serious error (for this to happen the software fails to respond in time). As a result the hardware ensures the CRC is incorrect at the end of the message, thus guaranteeing the message to be discarded at the receiver.
56.4.5 **UCPD BMC receiver**

The UCPD BMC receiver performs:

- Clock recovery
- Preamble detection / timing derivation
- BMC decoding
- 4b5b decoding
- K-code ordered set recognition
- CRC checking
- SOP detection
- EOP detection

The receiver is activated as soon as the UCPD peripheral is enabled (via UCPDEN), but it waits for an idle CC line state before attempting to receive a message.

The following figure shows the UCPD BMC receiver high-level architecture.

![UCPD BMC receiver architecture](image)

**CRC checker**

The received bits are fed into a CRC checker which evolves a 32-bit state during the received the payload bitstream. At the end the 32 bits of the CRC also fed into the logic.

The EOP detection (5 bits) halts the process and a check is performed for the fixed residual state which confirms correct reception of the payload (in fact the residual is 0xC704DD78).

At this point the UCPD raises interrupt RXMSGEND. If the CRC was not correct then RXERR is set true and the receive data must be discarded.

Under normal operation, this interrupt would previously have been acknowledged and thus cleared. If this is not the case, a different interrupt RXOVR is generated in place of RXMSGEND.
Ordered set detection

This function detects the different ordered sets each consisting of four 5-bit K-codes. Once we are in the preamble we open a sliding window detection of the ordered set (4 words of 5 bits).

The ordered sets detected include all SOP* codes (SOP, SOP', and SOP''), but also Hard Reset, Cable Reset, SOP'_Debug, SOP''_Debug, and two extensions defined by registers UCPD_RX_ORDEXT1 and UCPD_RX_ORDEXT2.

EOP detection and Hard Reset exception handling

EOP is a fixed 5-bit K-code marking the end of a message.

The way in which a transmitter is required to send a Hard Reset (if a previous message transmit is still in progress) is that this previous message is truncated early with an EOP.

If Hard Reset were ignored, then the EOP detection can be done only at the expected time. However, due to the Hard Reset issue, the EOP detector must be active while an Rx message is arriving. When an “early EOP” is detected, the truncated Rx message is immediately discarded.

Truncated or corrupted message exception

Once the ordered set has been detected, depending on the message, there may be data bytes to be received which is completed with a CRC and EOP. If at any point during these phases an error condition happens:

• the line becomes static for a time significantly longer than one “UI” period (the exact threshold for this condition is not critical but the exception must occur before three UIs), or
• the message goes to the end but it is not recognized (for example EOP is corrupted).

In both cases, the receiver quits the current message, raising RXMSGEND and RXERR flags.

Short preamble or incomplete ordered set exception

In the exceptional case of the receiver seeing less that half of the expected preamble, the frequency estimation allowing correct BMC-decode becomes impossible. Even if the full preamble is seen, allowing frequency estimation, but the ordered set is not fully received before the line becomes static, the receiver state machine does not start.

In both of these cases, the clock-recovery/BMC decoder re-starts, checking initially for an IDLE condition, followed by a preamble, and then estimating frequency.

56.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd)

UCPD offers simple control of these resistors via ANAMODE and ANASUBMODE[1:0]. In case only one of the CC lines is to be used, it is possible to optimize power consumption by disabling control on the other line, through the CCENABLE[1:0] bitfield.

When the MCU is unpowered, it still presents the “dead battery” Rd, provided that UCPDx_DBCC1 and UCPDx_DBCC2 pins are each connected to UCPDx_CC1 and UCPDx_CC2 pins, respectively.

If dead battery behavior is not required (for example for source only products), then UCPDx_DBCC1 and UCPDx_DBCC2 pins must both be tied to ground.
After power arrives and the MCU boots, the desired behavior (for example source) must be programmed into ANAMODE and ANASUBMODE[1:0] before setting the UCPD_DBDIS bit of the PWR_CR3 register to remove dead battery pull-down resistor and allow the values just programmed to take effect.

Use of Standby low-power mode is possible for sinks in the unattached state.

56.4.7 UCPD Type-C voltage monitoring and de-bouncing

For correct operation of the Type-C state machine and for detecting the cable orientation, the CC1/2 lines must be monitored for voltage level, while ignoring fast events such as peaks.

Thresholds between voltage levels on the CC1/2 lines are determined through PHY threshold detector settings.

The TYPEC_VSTATE_CC1/2[1:0] bitfields reflect the CC1/2 line levels processed with a hardware de-bouncing filter that suppresses high-speed line events such as peaks. The PHYCCSEL bit selects the line, CC1 or CC2, to be used for Power Delivery signaling.

For minimizing the power consumption, it is recommended to use the polling method, with the Type-C detectors only turned on for the instant of polling, rather than keeping the Type-C detectors permanently on and wake the device up from Stop mode upon CC1/2 line events.

56.4.8 UCPD fast role swap (FRS)

FRS signaling

The FRS condition (a pulse of a specific length), is generated upon setting the FRSTX bit.

For the duration of FRS condition, the I/O configured as UCPD_FRSTX (alternate function) controls, with high level, the gate of an external NMOS transistor that pulls the active CC line down.

FRS detection

FRS monitoring is enabled by setting the bit FRSRXEN, after writing PHYCCSEL that selects the active CC line depending on the cable orientation detected.

56.4.9 UCPD DMA Interface

DMA is implemented in the UCPD and when it is enabled the byte-level interrupts to handle UCPD1_TXDR and UCPD1_RXDR registers (Tx and Rx data register, each one byte) are no longer needed.

By enabling bits TXDMAEN and/or RXDMAEN, DMA can be activated independently for Tx and/or Rx functionality.

56.4.10 Wake-up from Stop mode

For power consumption optimization, it is useful to use Stop mode and wait for events on CC lines to wake the MCU up.

In order for this to work, it must be first enabled by writing a 1 to WUPEN.
The events causing the wake-up can be:

- Events on the BMC receiver (RXORDDET, RXHRSTDET), hardware enable PHYRXEN
- Event on the FRS detector (FRSEVT), hardware enable FRSRXEN
- Events on the Type-C detectors (TYPECEVT1, TYPECEVT2), hardware enables CC1TCDIS, CC2TCDIS

56.5 UCPD programming sequences

The normal sequence of use of the UCPD unit is:
1. Configure UCPD.
2. Enable UCPD.
3. Concurrently:
   - On demand from protocol layer, send Tx message
   - Intercept (poll or wait for interrupt) relevant Rx messages and recover detail to hand off to protocol layer

Repeat the last point infinitely.

56.5.1 Initialization phase

Use the following sequence for a clean startup:
1. Prepare all initial clock divider values, by writing the UCPD_CFG register.
2. Enable the unit, by setting the UCPDEN bit.
3. Enable the analog Rx filter of either CC line, via the RXAFILTEN bit of the UCPD_CFG2 register.

56.5.2 Type-C state machine handling

For the general application cases of source, sink, or dual-role port (the last alternating the source and the sink), the software must implement a corresponding USB Type-C state machine. The basic coding is in the following table.

<table>
<thead>
<tr>
<th>ANAMODE</th>
<th>ANASUBMODE[1:0]</th>
<th>Notes</th>
<th>TYPEC_VSTATE_CCx[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Source</td>
<td>00: Disabled</td>
<td></td>
<td>00</td>
</tr>
<tr>
<td></td>
<td>01: Default USB Rp</td>
<td>-</td>
<td>vRa[Def] vRd[Def] vOPEN[Def]</td>
</tr>
<tr>
<td></td>
<td>10: 1.5A Rp</td>
<td>-</td>
<td>vRa[1.5] vRd[1.5] vOPEN[1.5]</td>
</tr>
<tr>
<td>1: Sink</td>
<td>xx</td>
<td>-</td>
<td>vRa vRd-USB vRd-1.5 vRd-3.0</td>
</tr>
</tbody>
</table>

The CCENABLE[1:0] bitfield can disable pull-up/pull-downs on one of the CC lines.

Note: The Type-C state machine depends not only on CC line levels, but also on VBUS presence detection (sink mode) and, when in source mode, determines VCONN generation and
VBUS state (ON/OFF/+voltage level); discharge). UCPD does not directly control VBUS generation circuitry nor VCONN load switch (enabling VCONN supply generator to be connected to the CC line), but the application needs these inputs and controls, to function correctly.

General programming sequence (with UCPD configured then enabled)

1. Set ANAMODE and ANASUBMODE[1:0] based on the current position in USB Type-C state machine (and also the current advertisement in the case of a source). This turns on the appropriate pull-ups/pull-downs on the CC lines, and defines the voltage levels that the TYPEC_VSTATE fields represent. Note that before programming, the PHY is effectively off.

2. Read TYPEC_VSTATE_CC1/2 to determine the initial Type-C state (for example whether the local source is connected to a remote sink).

3. In the case of no connection, wait for a connection event.

4. Assuming a connection is detected and assuming a local Power Delivery functionality is implemented, start sending/receiving Power Delivery messages.

5. When a new interrupt/event occurs on PHYEV1/2 indicating a change in stable voltage, re-evaluate the implications and give this input to the Type-C state machine.

Case of a source that needs to change between one of the three possible Rp values (Default-USB / 1.5A / 3.0A) and the sink connected to it:

- [Source] Simply reprogram ANASUBMODE[1:0]
- [Sink behavior from that time] PHYEV1/2 occurs and the TYPEC_VSTATE1/2 changes accordingly

Programming for a dual-role port (DRP) toggling from source to sink:

- Simply re-program ANAMODE and ANASUBMODE[1:0] to start the new behavior

Detailed programming sequence (example):
### 56.5.3 USB PD transmit

On reception of a message from the protocol layer (that is, to be sent), prepare Tx message contents by writing the UCPD_TX_ORDSET and UCPD_TX_PAYSZ registers.

The message transmission is triggered by setting the TXSEND bit, with an appropriate value of the TXMODE bitfield.

When the data byte is transmitted, the TXIS flag is raised to request a new data write to the UCPD_TXDR register.

This re-iterates until the entire payload of data is transmitted.

---

#### Table 613. Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2)

<table>
<thead>
<tr>
<th>Type-C state</th>
<th>ANAMODE; ANASUBMODE DE[1:0]</th>
<th>CCENABLE</th>
<th>PHYCSEL</th>
<th>RDCH</th>
<th>CC[x]</th>
<th>VCONN EN[1]</th>
<th>Event =&gt; go to next line</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unattached. SRC</td>
<td>0:Source; 11:Rp3A0</td>
<td>11:both enabled</td>
<td>0 (don’t care)</td>
<td>00: [neither]</td>
<td>PHYEV1 1: [VRd-3A0]</td>
<td>Wait for sink attach detect; seen on CC1 [EVT1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attachwait. SRC</td>
<td>0:Source; 11:Rp3A0</td>
<td>11:both enabled</td>
<td>0 (don’t care)</td>
<td>00: [neither]</td>
<td>PHYEV2 2: [VRa]</td>
<td>Attachwait started (100-200 ms); now also see the Ra =&gt; requesting VCONN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attached. SRC [VCONN =&gt; CC2]</td>
<td>0:Source; 11:Rp3A0</td>
<td>11:both enabled</td>
<td>0 (don’t care)</td>
<td>00: [neither]</td>
<td>Timer (100 ms) and no PHYEV1</td>
<td>Local CC2 disconnected from PHY (VCONN switch connects VCONN source to CC2 externally; Continue to monitor PHYEV1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0:Source; 10:Rp1A5</td>
<td>[SinkTxOK]</td>
<td>01: CC2 disable (possible and recommend ed due to external VCONN switch)</td>
<td>10: [CC2 active]</td>
<td>PHYEV1: [VOpen-3A0]</td>
<td>Source wants to initiate message sequence (SinkTxNG condition set first)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0:Source; 11:Rp3A0</td>
<td>[SinkTxOK]</td>
<td>01: CC2 disable (possible and recommend ed due to external VCONN switch)</td>
<td>10: [CC2 active]</td>
<td>PHYEV1: [VOpen-3A0]</td>
<td>Source finished message sequence (SinkTxOK condition afterwards)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unattached wait. SRC</td>
<td>0:Source; 11:Rp3A0</td>
<td>11:both enabled</td>
<td>0 (do not care)</td>
<td>00: [neither]</td>
<td>&gt; 0.8V detection</td>
<td>Discharge VCONN [CC2] actively [Rdch]; to &lt; 0.8V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unattached. SRC</td>
<td>0:Source; 11:Rp3A0</td>
<td>11:both enabled</td>
<td>0 (do not care)</td>
<td>00: [neither]</td>
<td>&gt; 0.8V detection</td>
<td>Discharge VCONN [CC2] actively [Rdch]; to &lt; 0.8V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Two GPIOs to enable VCONN through external load switch components
Upon sending the CRC packet, the TXMSGSENT flag is set to indicate the completion of the message transmission.

**Hard Reset transmission**

As soon as it is known that a Hard Reset needs to be transmitted, setting the TXHRST bit of the UCPD_CR register forces the internal state machine to generate the correct sequence. The value of UCPD_TX_ORDSET does not require update in this precise case (the correct code for Hard Reset is sent by UCPD).

The USB Power Delivery specification requires that in the case of an ongoing message transmission, the Hard Reset takes precedence. In this case, for example, UCPD truncates the payload of the current message, appending EOP to the end. No notification is available via the registers (for example through the TXMSGSEND flag). This is justified by the fact that the Hard Reset takes precedence over any previous activity (for which it is therefore no longer important to know if it is completed).

**Use of DMA for transmission**

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the UCPD_CR register.

For each message:

- Prepare the whole message in memory (starting with two header bytes)
- Program the DMA operation with a length corresponding to the two header bytes plus a number of data bytes corresponding to the number of data words multiplied by four
- Write TXSEND to initiate the message transfer
- If TXMSGDISC then go back to previous line (TXSEND)
- Wait for DMA transfer complete interrupt (that is, when last Tx byte written to UCPD)
- Double-check subsequent TXMSGSENT interrupt appears

### 56.5.4 USB PD receive

Notification of start of the receive message sequence is triggered by an interrupt on UCPD_SR (bit RXORDDET).

The information is recovered by reading:

- UCPD_RX_SOP (on interrupt RXORDDET)
- UCPD_RXDR (on interrupt RXNE, repeats for each byte)
- UCPD_RXPAYSZ (on interrupt RXMSGEND)

The data previously read from UCPD_RXDR above must be discarded at this point if the RXERR flag is set.

If the CRC is valid, the received data is transferred to the protocol layer.

For debug purposes, it may be desirable to track statistics of the number of incorrect K-codes received (this is done only when 3/4 K-codes were valid as defined in the specification). This is facilitated through:

- RXSOP3OF4 bit indicating the presence of at least one invalid K-code
- RXSOPKINVALID bitfield identifying the order of invalid K-code in the ordered set
Use of DMA for reception

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the UCPD_CR register.

Whenever a Rx message is expected:
- Program a DMA receive operation (and spare buffer) a little longer than the maximum possible message (length depends on extended message support).
- After receiving RXORDDET, DMA operation starts working in the background.
- On reception of RXMSGEND interrupt, read RXPAYSZ.
- Double-check RXPAYSZ vs. the number of DMA Rx bytes (must correspond but DMA read of RXDR is slightly after RXDR gets last byte).
- Process the DMA Rx buffer.
- Prepare next Rx DMA buffer as soon as possible in order to be ready.

56.5.5 UCPD software trimming

The CC pull-up (Rp) and pull-down (Rd) devices must be trimmed on each part, to meet the required accuracy. The trimming values are saved in the non-volatile memory.

To trim the CC pull-up and pull-down devices by software, apply the following procedure:
1. Retrieve the trim values from the non-volatile memory (refer to Table 605: UCPD software trim data)
2. At initialization, write the trim values to the UCPD_CFGR3 register bitfields as follows:
   - 3A0_CC1[3:0] to TRIM_CC1_RP[3:0]
   - 3A0_CC2[3:0] to TRIM_CC2_RP[3:0]
   - Rd_CC1[3:0] to TRIM_CC1_RD[3:0]
   - Rd_CC2[3:0] to TRIM_CC2_RD[3:0]
3. At each setting of ANASUBMODE to 1A5 or 3A0, respectively, write the trimming values to the UCPD_CFGR3 register bitfields as follows:
   - 1A5_CC1[3:0] or 3A0_CC1[3:0], respectively, to TRIM_CC1_RP[3:0]
   - 1A5_CC2[3:0] or 3A0_CC2[3:0], respectively, to TRIM_CC2_RP[3:0]

56.6 UCPD low-power modes

A summary of low-power modes is shown below in Table 614: Effect of low power modes on the UCPD.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect</td>
</tr>
<tr>
<td>Stop</td>
<td>Detection of events (Type-C, BMC Rx, FRS detection) remains operational and can wake up the MCU.</td>
</tr>
<tr>
<td>Standby</td>
<td>UCPD is not operating, and cannot wake up the MCU. Pull-downs remain active if configured.</td>
</tr>
<tr>
<td>Unpowered</td>
<td>Dead battery pull-downs remain active.</td>
</tr>
</tbody>
</table>
The UCPD is able to wake up the MCU from Stop mode when it recognizes a relevant event, either:

- Type-C event relating to a change in the voltage range seen on either of the CC lines, visible in TYPEC_VSTATE_CCx
- Power delivery receive message with an ordered set matching those filtered according to RXORDSETEN[8:0], visible by reading RXORDSET

Wake-up from Stop mode is enabled by setting the WUPEN bit in the UCPD_CFG2 register.

At UCPD level three types of event requiring kernel clock activity may occur during Stop mode:

- Activity on the analog PHY voltage threshold detectors which can later be confirmed to be a stable change between voltage ranges defined in the Type-C specification
- Activity on Power Delivery BMC receiver (coming from the selected CC line) which can potentially generate an Rx message event (that is, RXORDSET) later
- Activity on Power Delivery FRS detector which can potentially generate an FRS signaling detection event (that is, FRSEVT) later

It order to function correctly with the RCC, the clock request signal is activated (conditional on WUPEN) when there is asynchronous activity on:

- Type-C voltage threshold detectors (coming from either CC line)
- Power Delivery receiver signal (from the selected CC line)
- FRS detection signal (from the selected CC line)

### 56.7 UCPD interrupts

The table below lists the UCPD event flags, with the associated flag clear bits and interrupt enable bits.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Event flag/Interrupt clearing method</th>
<th>Interrupt enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRS detection</td>
<td>FRSEVT</td>
<td>Set FRSEVTCF</td>
<td>FRSEVTIE</td>
</tr>
<tr>
<td>Type C voltage level change on CC2</td>
<td>TYPECEVT2</td>
<td>Set TYPECEVT2CF</td>
<td>TYPECEVT2IE</td>
</tr>
<tr>
<td>Type C voltage level change on CC1</td>
<td>TYPECEVT1</td>
<td>Set TYPECEVT1CF</td>
<td>TYPECEVT1IE</td>
</tr>
<tr>
<td>Rx message received</td>
<td>RXMSGSEND</td>
<td>Set RXMSGENDCF</td>
<td>RXMSGENDIE</td>
</tr>
<tr>
<td>Rx data overflow</td>
<td>RXOVR</td>
<td>Set RXOVRCF</td>
<td>RXOVR</td>
</tr>
<tr>
<td>Rx Hard Reset detected</td>
<td>RXHRSTDET</td>
<td>Set RXHRSTDETCF</td>
<td>RXHRSTDETCIE</td>
</tr>
<tr>
<td>Rx ordered set (4 K-codes) detected</td>
<td>RXORDDET</td>
<td>Set RXORDDETCF</td>
<td>RXORDDETCIE</td>
</tr>
<tr>
<td>Receive data register not empty</td>
<td>RXNE</td>
<td>Read data in UCPD_RXDR</td>
<td>RXNEIE</td>
</tr>
<tr>
<td>Tx data underrun</td>
<td>TXUND</td>
<td>Set TXUNDCF</td>
<td>TXUNDIE</td>
</tr>
<tr>
<td>Hard Reset sent</td>
<td>HRSTSENT</td>
<td>Set HRSTSENTCF</td>
<td>HRSTSENTIE</td>
</tr>
<tr>
<td>Hard Reset discarded</td>
<td>HRSTDISC</td>
<td>Set HRSTDISCCF</td>
<td>HRSTDISCIE</td>
</tr>
<tr>
<td>Transmit message aborted</td>
<td>TXMSGABT</td>
<td>Set TXMSGABTCF</td>
<td>TXMSGABTIE</td>
</tr>
</tbody>
</table>
When an interrupt from the UCPD is received, then the software has to check what is the source of the interrupt by reading the UCPD_SR register. Depending on which bit is at 1, the ISR must handle that condition and clear the bit by a write to the appropriate bit of the UCPD_ICR register.

### 56.8 UCPD registers

#### 56.8.1 UCPD configuration register 1 (UCPD_CFGR1)

**Address offset:** 0x000  
**Reset value:** 0x0000 0000  

General configuration of the peripheral. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15 14</td>
<td>13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bit 31** **UCPDEN**: UCPD peripheral enable  
General enable of the UCPD peripheral.  
0: Disable  
1: Enable  
Upon disabling, the peripheral instantly quits any ongoing activity and all control bits and bitfields default to their reset values. They must be set to their desired values each time the peripheral transits from disabled to enabled state.

**Bit 30** **RXDMAEN**: Reception DMA mode enable  
When set, the bit enables DMA mode for reception.  
0: Disable  
1: Enable
Bit 29 TXDMAEN: Transmission DMA mode enable
 When set, the bit enables DMA mode for transmission.
 0: Disable
 1: Enable

Bits 28:20 RXORDSETEN[8:0]: Receiver ordered set enable
 The bitfield determines the types of ordered sets that the receiver must detect. When
 set/cleared, each bit enables/disables a specific function:
 0bXXXXXXXX1: SOP detect enabled
 0bXXXXXXXX1X: SOP’ detect enabled
 0bXXXXXXXX1XX: SOP” detect enabled
 0bXXXXXX1XXXX: Hard Reset detect enabled
 0bXXXXX1XXXX: Cable Detect reset enabled
 0bXXXX1XXXXX: SOP_Debug enabled
 0bXX1XXXXXX: SOP”_Debug enabled
 0bX1XXXXXXX: SOP extension#1 enabled
 0b1XXXXXXXX: SOP extension#2 enabled

Bits 19:17 PSC_USBPDCLK[2:0]: Pre-scaler division ratio for generating ucpd_clk
 The bitfield determines the division ratio of a kernel clock pre-scaler producing UCPD
 peripheral clock (ucpd_clk).
 0x0: 1 (bypass)
 0x1: 2
 0x2: 4
 0x3: 8
 0x4: 16
 It is recommended to use the pre-scaler so as to set the ucpd_clk frequency in the range
 from 6 to 9 MHz.

Bit 16 Reserved, must be kept at reset value.

Bits 15:11 TRANSWIN[4:0]: Transition window duration
 The bitfield determines the division ratio (the bitfield value minus one) of a hbit_clk divider
 producing tTransitionWindow interval.
 0x0: Not supported
 0x01: 2
 0x09: 10 (recommended)
 0x1F: 32
 Set a value that produces an interval of 12 to 20 us, taking into account the ucpd_clk
 frequency and the HBITCLKDIV[5:0] bitfield setting.

Bits 10:6 IFRGAP[4:0]: Division ratio for producing inter-frame gap timer clock
 The bitfield determines the division ratio (the bitfield value minus one) of a ucpd_clk divider
 producing inter-frame gap timer clock (tInterFrameGap).
 0x0: Not supported
 0x01: 2
 0x0D: 14
 0x0E: 15
 0x0F: 16
 0x1F: 32
 The division ratio 15 is to apply for Tx clock at the USB PD 2.0 specification nominal value.
 The division ratios below 15 are to apply for Tx clock below nominal, and the division ratios
 above 15 for Tx clock above nominal.
### UCPD configuration register 2 (UCPD_CFGR2)

Address offset: 0x004  
Reset value: 0x0000 0000

Configuration of the UCPD Rx signal filtering. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bypass</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: 27</td>
<td>2: 64</td>
<td>3: 64</td>
<td>4: 64</td>
<td>5: 64</td>
<td>6: 64</td>
<td>7: 64</td>
<td>8: 64</td>
</tr>
<tr>
<td>9: 64</td>
<td>10: 64</td>
<td>11: 64</td>
<td>12: 64</td>
<td>13: 64</td>
<td>14: 64</td>
<td>15: 64</td>
<td>16: 64</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RXAFIL TEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:9** Reserved, must be kept at reset value.

**Bit 8** RXAFILTEN: Rx analog filter enable  
Setting the bit enables the Rx analog filter required for optimum Power Delivery reception.  
0: Disable  
1: Enable

**Bits 7:4** Reserved, must be kept at reset value.

**Bit 3** WUPEN: Wake-up from Stop mode enable  
Setting the bit enables the UCPD_ASYNC_INT signal.  
0: Disable  
1: Enable

**Bit 2** FORCECLK: Force ClkReq clock request  
0: Do not force clock request  
1: Force clock request

**Bit 1** RXFILT2N3: BMC decoder Rx pre-filter sampling method  
Number of consistent consecutive samples before confirming a new value.  
0: 3 samples  
1: 2 samples

**Bit 0** RXFILTDIS: BMC decoder Rx pre-filter enable  
0: Enable  
1: Disable

The sampling clock is that of the receiver (that is, after pre-scaler).
56.8.3 UCPD configuration register 3 (UCPD_CFGR3)

Address offset: 0x008
Reset value: 0x0000 0000

Configuration of UCPD trimming of the CC pull-up and pull-down devices. The trimming is managed by hardware until the first software write into this register.

The register is reserved (must not be written) for devices that support the fully automatic trimming. Refer to Table 604: UCPD implementation.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Res.</td>
</tr>
<tr>
<td>30</td>
<td>Res.</td>
</tr>
<tr>
<td>29</td>
<td>Res.</td>
</tr>
<tr>
<td>28</td>
<td>Res.</td>
</tr>
<tr>
<td>27</td>
<td>TRIM_CC2_RP[3:0]</td>
</tr>
<tr>
<td>26</td>
<td>Res.</td>
</tr>
<tr>
<td>25</td>
<td>Res.</td>
</tr>
<tr>
<td>24</td>
<td>Res.</td>
</tr>
<tr>
<td>23</td>
<td>Res.</td>
</tr>
<tr>
<td>22</td>
<td>Res.</td>
</tr>
<tr>
<td>21</td>
<td>TRIM_CC2_RD[3:0]</td>
</tr>
<tr>
<td>20</td>
<td>Res.</td>
</tr>
<tr>
<td>19</td>
<td>Res.</td>
</tr>
<tr>
<td>18</td>
<td>Res.</td>
</tr>
<tr>
<td>17</td>
<td>Res.</td>
</tr>
<tr>
<td>16</td>
<td>Res.</td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 TRIM_CC2_RP[3:0]: SW trim value for Rp current sources on the CC2 line

Bits 24:20 Reserved, must be kept at reset value.

Bits 19:16 TRIM_CC2_RD[3:0]: SW trim value for Rd resistor on the CC2 line

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:9 TRIM_CC1_RP[3:0]: SW trim value for Rp current sources on the CC1 line

Bits 8:4 Reserved, must be kept at reset value.

Bits 3:0 TRIM_CC1_RD[3:0]: SW trim value for Rd resistor on the CC1 line

56.8.4 UCPD control register (UCPD_CR)

Address offset: 0x00C
Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).
Bits 31:22  Reserved, must be kept at reset value.

Bit 21  **CC2TCDIS**: CC2 Type-C detector disable
   The bit disables the Type-C detector on the CC2 line.
   0: Enable
   1: Disable
   When enabled, the Type-C detector for CC2 is configured through ANAMODE and
   ANASUBMODE[1:0].

Bit 20  **CC1TCDIS**: CC1 Type-C detector disable
   The bit disables the Type-C detector on the CC1 line.
   0: Enable
   1: Disable
   When enabled, the Type-C detector for CC1 is configured through ANAMODE and
   ANASUBMODE[1:0].

Bit 19  Reserved, must be kept at reset value.

Bit 18  **RDCH**: Rdch condition drive
   The bit drives Rdch condition on the CC line selected through the PHYCCSEL bit (thus
   associated with VCONN), by remaining set during the source-only UnattachedWait.SRC
   state, to respect the Type-C state. Refer to "USB Type-C ECN for Source VCONN
   Discharge". The CCENABLE[1:0] bitfield must be set accordingly, too.
   0: No effect
   1: Rdch condition drive

Bit 17  **FRSTX**: FRS Tx signaling enable.
   Setting the bit enables FRS Tx signaling.
   0: No effect
   1: Enable
   The bit is cleared by hardware after a delay respecting the USB Power Delivery specification
   Revision 3.1.

Bit 16  **FRSRXEN**: FRS event detection enable
   Setting the bit enables FRS Rx event (FRSEVT) detection on the CC line selected through
   the PHYCCSEL bit. 0: Disable
   1: Enable
   Clear the bit when the device is attached to an FRS-incapable source/sink.

Bit 15  Reserved, must be kept at reset value.

Bit 14  Reserved, must be kept at reset value.

Bit 13  Reserved, must be kept at reset value.

Bit 12  Reserved, must be kept at reset value.

Bits 11:10  **CCENABLE[1:0]**: CC line enable
   This bitfield enables CC1 and CC2 line analog PHYs (pull-ups and pull-downs) according to
   ANAMODE and ANASUBMODE[1:0] setting.
   0x0: Disable both PHYs
   0x1: Enable CC1 PHY
   0x2: Enable CC2 PHY
   0x3: Enable CC1 and CC2 PHY
   A single line PHY can be enabled when, for example, the other line is driven by VCONN via
   an external VCONN switch. Enabling both PHYs is the normal usage for sink/source.
Bit 9 **ANAMODE**: Analog PHY operating mode
0: Source
1: Sink
The use of CC1 and CC2 depends on CCENABLE. Refer to Table 612: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx for the effect of this bitfield in conjunction with ANASUBMODE[1:0].

Bits 8:7 **ANASUBMODE[1:0]**: Analog PHY sub-mode
Refer to Table 612: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx for the effect of this bitfield.

Bit 6 **PHYCCSEL**: CC1/CC2 line selector for USB Power Delivery signaling
0: Use CC1 IO for Power Delivery communication
1: Use CC2 IO for Power Delivery communication
The selection depends on the cable orientation as discovered at attach.

Bit 5 **PHYRXEN**: USB Power Delivery receiver enable
0: Disable
1: Enable
Both CC1 and CC2 receivers are disabled when the bit is cleared. Only the CC receiver selected via the PHYCCSEL bit is enabled when the bit is set.

Bit 4 **RXMODE**: Receiver mode
Determines the mode of the receiver.
0: Normal receive mode
1: BIST receive mode (BIST test data mode)
When the bit is set, RXORDSET behaves normally, RXDR no longer receives bytes yet the CRC checking still proceeds as for a normal message. As this mode prevents reception of the header (containing MessageID), software has to auto-increment a received MessageID counter for inclusion in the GoodCRC acknowledge that must still be transmitted during this test.

Bit 3 **TXHRST**: Command to send a Tx Hard Reset
0: No effect
1: Start Tx Hard Reset message
The bit is cleared by hardware as soon as the message transmission begins or is discarded.

Bit 2 **TXSEND**: Command to send a Tx packet
0: No effect
1: Start Tx packet transmission
The bit is cleared by hardware as soon as the packet transmission begins or is discarded.

Bits 1:0 **TXMODE[1:0]**: Type of Tx packet
Writing the bitfield triggers the action as follows, depending on the value:
0x0: Transmission of Tx packet previously defined in other registers
0x1: Cable Reset sequence
0x2: BIST test sequence (BIST Carrier Mode 2)
Others: invalid
From V1.1 of the USB PD specification, there is a counter defined for the duration of the BIST Carrier Mode 2. To quit this mode correctly (after the "tBISTContMode" delay), disable the peripheral (UCPDEN = 0).
56.8.5 UCPD interrupt mask register (UCPD_IMR)

Address offset: 0x010
Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td><strong>TYPECEVT2IE</strong></td>
<td><strong>TYPECEVT1IE</strong></td>
<td>Res.</td>
<td><strong>RXMSGENDIE</strong></td>
<td><strong>RXOVRIE</strong></td>
<td><strong>RXHRSTDETIE</strong></td>
<td><strong>RXORDDETIE</strong></td>
<td><strong>RXNEIE</strong></td>
<td><strong>TXUNDIE</strong></td>
<td><strong>HRSTSENTIE</strong></td>
<td><strong>HRSTDISCIE</strong></td>
<td><strong>TXMSGABTE</strong></td>
<td><strong>TXMSGBBTIE</strong></td>
<td><strong>TXMSGDETIE</strong></td>
<td><strong>TXISIE</strong></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVTIE**: FRSEVT interrupt enable
0: Disable
1: Enable

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **TYPECEVT2IE**: TYPECEVT2 interrupt enable
0: Disable
1: Enable

Bit 14 **TYPECEVT1IE**: TYPECEVT1 interrupt enable

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RXMSGENDIE**: RXMSGEND interrupt enable
0: Disable
1: Enable

Bit 11 **RXOVRIE**: RXOVRIE interrupt enable
0: Disable
1: Enable

Bit 10 **RXHRSTDETIE**: RXHRSTDET interrupt enable
0: Disable
1: Enable

Bit 9 **RXORDDETIE**: RXORDDET interrupt enable
0: Disable
1: Enable

Bit 8 **RXNEIE**: RXNEIE interrupt enable
0: Disable
1: Enable

Bit 7 Reserved, must be kept at reset value.
Bit 6 **TXUNDIE**: TXUND interrupt enable  
0: Disable  
1: Enable

Bit 5 **HRSTSENTE**: HRSTSENT interrupt enable  
0: Disable  
1: Enable

Bit 4 **HRSTDISCIE**: HRSTDISC interrupt enable  
0: Disable  
1: Enable

Bit 3 **TXMSGABTIE**: TXMSGABT interrupt enable  
0: Disable  
1: Enable

Bit 2 **TXMSGSENTIE**: TXMSGSENT interrupt enable  
0: Disable  
1: Enable

Bit 1 **TXMSGDISCIE**: TXMSGDISC interrupt enable  
0: Disable  
1: Enable

Bit 0 **TXISIE**: TXIS interrupt enable  
0: Disable  
1: Enable

### 56.8.6 UCPD status register (UCPD_SR)

Address offset: 0x014  
Reset value: 0x0000 0000

The flags (single-bit status bitfields) are associated with interrupt request. Interrupt is generated if enabled by the corresponding bit of the UCPD_IMR register.

<table>
<thead>
<tr>
<th>Bit 31:21 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:21 Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31:21 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:21 Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bit 20 **FRSEVT**: FRS detection event

The flag is cleared by setting the FRSEVTCF bit.
0: No new event
1: New FRS receive event occurred

Bits 19:18 **TYPEC_VSTATE_CC2[1:0]**: CC2 line voltage level

The status bitfield indicates the voltage level on the CC2 line in its steady state.
0x0: Lowest
0x1: Low
0x2: High
0x3: Highest

The voltage variation on the CC2 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bits 17:16 **TYPEC_VSTATE_CC1[1:0]**:

The status bitfield indicates the voltage level on the CC1 line in its steady state.
0x0: Lowest
0x1: Low
0x2: High
0x3: Highest

The voltage variation on the CC1 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bit 15 **TYPECEVT2**: Type-C voltage level event on CC2 line

The flag indicates a change of the TYPEC_VSTATE_CC2[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2CF bit.
0: No new event
1: A new Type-C event

Bit 14 **TYPECEVT1**: Type-C voltage level event on CC1 line

The flag indicates a change of the TYPEC_VSTATE_CC1[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2CF bit.
0: No new event
1: A new Type-C event

Bit 13 **RXERR**: Receive message error

The flag indicates errors of the last Rx message declared (via RXMSGEND), such as incorrect CRC or truncated message (a line becoming static before EOP is met). It is asserted whenever the RXMSGEND flag is set.
0: No error detected
1: Error(s) detected

Bit 12 **RXMSGEND**: Rx message received

The flag indicates whether a message (except Hard Reset message) has been received, regardless the CRC value. The flag is cleared by setting the RXMSGENDCF bit.
0: No new Rx message received
1: A new Rx message received

The RXERR flag set when the RXMSGEND flag goes high indicates errors in the last-received message.

Bit 11 **RXOVR**: Rx data overflow detection

The flag indicates Rx data buffer overflow. It is cleared by setting the RXOVRRCF bit.
0: No overflow
1: Overflow

The buffer overflow can occur if the received data are not read fast enough.
Bit 10  **RXHRSTDET**: Rx Hard Reset receipt detection
   The flag indicates the receipt of valid Hard Reset message. It is cleared by setting the RXHRSTDETCF bit.
   0: Hard Reset not received
   1: Hard Reset received

Bit 9  **RXORDDET**: Rx ordered set (4 K-codes) detection
   The flag indicates the detection of an ordered set. The relevant information is stored in the RXORDSET[2:0] bitfield of the UCPD_RX_ORDSET register. It is cleared by setting the RXORDDETCF bit.
   0: No ordered set detected
   1: A new ordered set detected

Bit 8  **RXNE**: Receive data register not empty detection
   The flag indicates that the UCPD_RXDR register is not empty. It is automatically cleared upon reading UCPD_RXDR.
   0: Rx data register empty
   1: Rx data register not empty

Bit 7  Reserved, must be kept at reset value.

Bit 6  **TXUND**: Tx data underrun detection
   The flag indicates that the Tx data register (UCPD_TXDR) was not written in time for a transmit message to execute normally. It is cleared by setting the TXUNDCF bit.
   0: No Tx data underrun detected
   1: Tx data underrun detected

Bit 5  **HRSTSENT**: Hard Reset message sent
   The flag indicates that the Hard Reset message is sent. The flag is cleared by setting the HRSTSENCF bit.
   0: No Hard Reset message sent
   1: Hard Reset message sent

Bit 4  **HRSTDISC**: Hard Reset discarded
   The flag indicates that the Hard Reset message is discarded. The flag is cleared by setting the HRSTDISCCF bit.
   0: No Hard Reset discarded
   1: Hard Reset discarded

Bit 3  **TXMSGABT**: Transmit message abort
   The flag indicates that a Tx message is aborted due to a subsequent Hard Reset message send request taking priority during transmit. It is cleared by setting the TXMSGABTCF bit.
   0: No transmit message abort
   1: Transmit message abort

Bit 2  **TXMSGSENT**: Message transmission completed
   The flag indicates the completion of packet transmission. It is cleared by setting the TXMSGSENCF bit.
   0: No Tx message completed
   1: Tx message completed

In the event of a message transmission interrupted by a Hard Reset, the flag is not raised.
56.8.7 UCPD interrupt clear register (UCPD_ICR)

Address offset: 0x018

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

| Bit 31:21 Reserved, must be kept at reset value. |
| Bit 20 | FRSEVTCF: FRS event flag (FRSEVT) clear |
| Setting the bit clears the FRSEVT flag in the UCPD_SR register. |
| Bit 19:16 Reserved, must be kept at reset value. |
| Bit 15 | TYPECEVT2CF: Type-C CC2 line event flag (TYPECEVT2) clear |
| Setting the bit clears the TYPECEVT2 flag in the UCPD_SR register. |
| Bit 14 | TYPECEVT1CF: Type-C CC1 event flag (TYPECEVT1) clear |
| Setting the bit clears the TYPECEVT1 flag in the UCPD_SR register. |
| Bit 13 Reserved, must be kept at reset value. |
| Bit 12 | RXMSGENDCF: Rx message received flag (RXMSGEND) clear |
| Setting the bit clears the RXMSGEND flag in the UCPD_SR register. |
Bit 11 **RXOVRCF**: Rx overflow flag (RXOVR) clear
Setting the bit clears the RXOVR flag in the UCPD_SR register.

Bit 10 **RXHRSTDETFC**: Rx Hard Reset detect flag (RXHRSTDET) clear
Setting the bit clears the RXHRSTDET flag in the UCPD_SR register.

Bit 9 **RXORDDETCF**: Rx ordered set detect flag (RXORDDET) clear
Setting the bit clears the RXORDDET flag in the UCPD_SR register.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TXUNDCF**: Tx underflow flag (TXUND) clear
Setting the bit clears the TXUND flag in the UCPD_SR register.

Bit 5 **HRSTSENDCF**: Hard reset send flag (HRSTSENT) clear
Setting the bit clears the HRSTSENT flag in the UCPD_SR register.

Bit 4 **HRSTDISCCF**: Hard reset discard flag (HRSTDISC) clear
Setting the bit clears the HRSTDISC flag in the UCPD_SR register.

Bit 3 **TXMSGABTCF**: Tx message abort flag (TXMSGABT) clear
Setting the bit clears the TXMSGABT flag in the UCPD_SR register.

Bit 2 **TXMSGSENCF**: Tx message send flag (TXMSGSENT) clear
Setting the bit clears the TXMSGSENT flag in the UCPD_SR register.

Bit 1 **TXMSGDSCCF**: Tx message discard flag (TXMSGDSC) clear
Setting the bit clears the TXMSGDSC flag in the UCPD_SR register.

Bit 0 Reserved, must be kept at reset value.

### 56.8.8 UCPD Tx ordered set type register (UCPD_TX_ORDSETR)

Address offset: 0x01C
Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1) and no packet transmission is in progress (TXSEND and TXHRST bits are both low).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXORDSET[19:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXORDSET[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **TXORDSET[19:0]**: Ordered set to transmit

The bitfield determines a full 20-bit sequence to transmit, consisting of four K-codes, each of five bits, defining the packet to transmit. The bit 0 (bit 0 of K-code1) is the first, the bit 19 (bit 4 of K-code4) the last.
56.8.9 UCPD Tx payload size register (UCPD_TX_PAYSZR)

Address offset: 0x020

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

<table>
<thead>
<tr>
<th>bits</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXPAYSZ[9:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TXPAYSZ[9:0]**: Payload size yet to transmit

The bitfield is modified by software and by hardware. It contains the number of bytes of a payload (including header but excluding CRC) yet to transmit: each time a data byte is written into the UCPD_TXDR register, the bitfield value decrements and the TXIS bit is set, except when the bitfield value reaches zero. The enumerated values are standard payload sizes before the start of transmission.

- 0x2: 2 bytes - the size of Control message from the protocol layer
- 0x6: 6 bytes - the shortest Data message allowed from the protocol layer
- 0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer
- 0x106: 262 bytes - the longest possible extended message
- 0x3FF: 1024 bytes - the longest possible payload (for future expansion)

56.8.10 UCPD Tx data register (UCPD_TXDR)

Address offset: 0x024

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

<table>
<thead>
<tr>
<th>bits</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXDATA[7:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]**: Data byte to transmit
56.8.11  UCPD Rx ordered set register (UCPD_RX_ORDSETR)

Address offset: 0x028
Reset value: 0x0000 0000

| Bits 31:7 | Reserved, must be kept at reset value. |
| Bits 6:4 | \texttt{RXSOPKINVALID[2:0]}: |
|          | The bitfield is for debug purposes only. |
|          | 0x0: No K-code corrupted |
|          | 0x1: First K-code corrupted |
|          | 0x2: Second K-code corrupted |
|          | 0x3: Third K-code corrupted |
|          | 0x4: Fourth K-code corrupted |
|          | Others: Invalid |
| Bit 3   | \texttt{RXSOP3OF4}: |
|         | The bit indicates the number of correct K-codes. For debug purposes only. |
|         | 0: 4 correct K-codes out of 4 |
|         | 1: 3 correct K-codes out of 4 |
| Bits 2:0 | \texttt{RXORDSET[2:0]}: Rx ordered set code detected |
|          | 0x0: SOP code detected in receiver |
|          | 0x1: SOP' code detected in receiver |
|          | 0x2: SOP'' code detected in receiver |
|          | 0x3: SOP'_Debug detected in receiver |
|          | 0x4: SOP''_Debug detected in receiver |
|          | 0x5: Cable Reset detected in receiver |
|          | 0x6: SOP extension#1 detected in receiver |
|          | 0x7: SOP extension#2 detected in receiver |
56.8.12 UCPD Rx payload size register (UCPD_RX_PAYSZR)

Address offset: 0x02C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 RXPAYSZ[9:0]: Rx payload size received

This bitfield contains the number of bytes of a payload (including header but excluding CRC) received: each time a new data byte is received in the UCPD_RXDR register, the bitfield value increments and the RXMSGEND flag is set (and an interrupt generated if enabled).

0x2: 2 bytes - the size of Control message from the protocol layer
0x6: 6 bytes - the shortest Data message allowed from the protocol layer
0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer
0x106: 262 bytes - the longest possible extended message
0x3FF: 1024 bytes - the longest possible payload (for future expansion)

The bitfield may return a spurious value when a byte reception is ongoing (the RXMSGEND flag is low).

56.8.13 UCPD receive data register (UCPD_RXDR)

Address offset: 0x030
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 RXDATA[7:0]: Data byte received
### 56.8.14 UCPD Rx ordered set extension register 1 (UCPD_RX_ORDEXTR1)

Address offset: 0x034  
Reset value: 0x0000 0000  

Writing to this register is only effective when the peripheral is disabled (UCPDEN = 0).

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit 31:20</th>
<th>Bit 19:0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reserved</td>
<td>RXSOPX1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[19:16]</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **RXSOPX1[19:0]**: Ordered set 1 received  
The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.

### 56.8.15 UCPD Rx ordered set extension register 2 (UCPD_RX_ORDEXTR2)

Address offset: 0x038  
Reset value: 0x0000 0000  

Writing to this register is only effective when the peripheral is disabled (UCPDEN = 0).

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit 31:20</th>
<th>Bit 19:0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reserved</td>
<td>RXSOPX2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[19:16]</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **RXSOPX2[19:0]**: Ordered set 2 received  
The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.
### 56.8.16 UCPD register map

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | UCPD_CFGR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | UCPD_CFGR2    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | UCPD_CFGR3    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | UCPD_CR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | UCPD_IMR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | UCPD_SR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
Refer to Section 2.3 on page 111 for the register boundary addresses.
57 Ethernet (ETH): media access control (MAC) with DMA controller

57.1 Ethernet introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

The Ethernet peripheral enables to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2008 standard.

The peripheral is configurable to meet the needs of a large variety of consumer and industrial applications.

57.2 Ethernet main features

The Ethernet peripheral embeds a dedicated DMA for direct memory interface, a media access controller (MAC) and a PHY interface block supporting several formats.

57.2.1 Standard compliance

The Ethernet peripheral is compliant with the following standards:
- IEEE 802.3-2008 for Ethernet MAC and media independent interface (MII)
- IEEE 1588-2008 for precision networked clock synchronization (PTP)
- IEEE 802.3az-2010 for Energy Efficient Ethernet (EEE)
- AMBA 2.0 for AHB master and AHB slave ports
- RMII specification version 1.2 from RMII consortium

57.2.2 MAC features

MAC Tx and Rx common features
- Separate transmission, reception, and control interfaces to the application
- 10, 100 Mbps data transfer rates with the following PHY interfaces:
  - IEEE 802.3-compliant MII interface to communicate with an external Fast Ethernet PHY
  - RMII interface to communicate with an external Fast Ethernet PHY
- Half-duplex operation:
  - CSMA/CD protocol support
  - Flow control using backpressure (based on implementation-specific white papers and UNH Ethernet Clause 4 MAC Test Suite - Annex D)
- Standard IEEE 802.3az-2010 for Energy Efficient Ethernet in MII PHYs
• 32-bit data transfer interface on the application side
• Full-duplex flow control operations (IEEE 802.3x Pause packets and Priority flow control)
• Network statistics with RMON or MIB counters (partial support of RFC2819/RFC2665)
• Ethernet packet timestamping as described in IEEE 1588-2002 and IEEE 1588-2008 (64-bit timestamps given in the Tx or Rx status of PTP packet). Both one-step and two-step timestamping are supported in Tx direction.
• Flexibility to control pulse-per-second (PPS) output signal (eth_ptp_pps_out and ETH_PPS_OUT)
• MDIO (Clause 22 and Clause 45) master interface for PHY device configuration and management

**MAC Tx features**
• Preamble and start-of-frame data (SFD) insertion
• Separate 32-bit status for each packet transmitted from the application
• Automatic CRC and pad generation controllable on a per-frame basis
• Programmable packet length to support Standard or Jumbo Ethernet packets of up to 16 Kbytes
• Programmable Inter Packet Gap (40–96 bit times in steps of 8)
• IEEE 802.3x Flow Control automatic transmission of zero-quanta Pause packet when flow control input transitions from assertion to de-assertion (in Full-duplex mode)
• Source address field insertion or replacement, and VLAN insertion, replacement, and deletion in transmitted packets with per-packet or static-global control
• Insertion, replacement, or deletion of up to two VLAN tags
• Option to transmit packets with reduced preamble size in Full-duplex mode
• Insert, replace, or delete queue/channel-based VLAN tags

**MAC Rx features**
• Automatic Pad and CRC stripping options
• Option to disable automatic CRC checking
• Preamble and SFD deletion
• Separate 112-bit or 128-bit status
• Programmable watchdog timeout limit
• Flexible address filtering modes:
  – Four 48-bit perfect (DA) address filters with masks for each byte
  – Four 48-bit SA address comparison check with masks for each byte
  – 64 bit Hash filter for multicast and unicast (DA) addresses
• Option to pass all multicast addressed packets
• Promiscuous mode to pass all packets without any filtering for network monitoring
• Pass all incoming packets (as per filter) with a status report
Additional packet filtering:
- VLAN tag-based: Perfect match and Hash-based filtering based either on the outer or inner VLAN tag
- Layer 3 and Layer 4-based: TCP or UDP over IPv4 or IPv6

IEEE 802.1Q VLAN tag detection and option to delete the VLAN tags in received packets

Detection of remote wake-up packets and AMD magic packets

Optional forwarding of received Pause packets to the application (in Full-duplex mode)

Layer 3/Layer 4 checksum offload for received packets

Stripping of up to two VLAN tags and providing the tags in the status

57.2.3 Transaction layer (MTL) features

MTL Tx and Rx Common Features
- 32-bit Transaction Layer block (bridges the application and the MAC)
- Optimization for packet-oriented transfers with packets delimiters
- Programmable burst length, up to half the size of the MTL Rx queue or Tx queue size, to support burst data transfer in the EQOS-MTL configuration
- Programmable threshold capability for each queue (default of 64 bytes)

MTL Tx features
- 2048-byte Transmit FIFO with programmable threshold capability
- Store-and-forward mechanism or threshold mode (cut-through) for transmission to the MAC
- Automatic retransmission of collision packets in Half-duplex mode
- Discard packets on late collision, excessive collisions, excessive deferral, and under-run conditions with appropriate status
- Module to calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum on frames transmitted in Store-and-forward mode
- Statistics by generating pulses for packets dropped (because of underflow) in the Tx FIFO
- Packet-level control for
  - VLAN tag insertion or replacement
  - Ethernet source address insertion
  - Layer 3/Layer 4 checksum insertion control
  - One-step timestamp
  - Timestamp control
  - CRC and pad control
MTL Rx features

- 2048-byte Receive FIFO with configurable threshold
- Programmable Rx queue threshold (default fixed at 64 bytes) in Threshold (or cut-through) mode
- Option to filter all error packets on reception and not forward them to the application in the store-and-forward mode
- Option to forward the undersized good packets
- Statistics by generating pulses for packets dropped (because of overflow) in the Rx FIFO
- Automatic generation of Pause packet control or backpressure signal to the MAC based on the Rx Queue fill level

57.2.4 DMA block features

The DMA block exchanges data between the peripheral and the system memory. DMA transfers are driven by software descriptors structure. The application can use a set of registers (see Section 57.11.2: Ethernet DMA registers) to control the DMA operations. The DMA block supports the following features:

- 32-bit data transfers
- Separate DMA in Transmit path and receive paths
- Optimization for packet-oriented DMA transfers with packet delimiters
- Byte-aligned addressing for data buffer support
- Dual-buffer (ring) descriptor support
- Descriptor architecture allowing large blocks of data transfer with minimum CPU intervention (each descriptor can transfer up to 32 Kbytes of data)
- Comprehensive status reporting normal operation and transfer errors
- Individual programmable burst length for Tx DMA and Rx DMA engines for optimal host bus utilization
- Programmable interrupt options for different operational conditions
- Per-packet Transmit or Receive Complete Interrupt control
- Round-robin or fixed-priority arbitration between the Receive and Transmit engines
- Start and Stop modes
- Separate ports for host control (AHB) access and host data interface
- Tx DMA channel with TCP segmentation offload (TSO) feature enabled
- Programmable control for Transmit Descriptor posted writes to improve the throughput

57.2.5 Bus interface features

AHB master interface

The AHB master interface features are the following:

- Interfaces with the application through AHB
- 32-bit data on the AHB master port
- Split, Retry, and Error AHB responses
- AHB 1-Kbyte boundary burst splitting
- Software-selected type of AHB burst (fixed burst, indefinite burst, or mix of both)
The AHB master interface does not generate the following:
- Wrap burst
- Locked or protected transfers

**AHB slave interface**

The AHB slave interface supports the following features:
- Interfaces with the application through AHB
- AHB slave interface (32-bit) for CSR access
- All AHB burst types

The AHB slave interface does not generate the following responses:
- Split
- Retry
- Error

### 57.3 Ethernet pins and internal signals

*Table 617* lists the Ethernet inputs and output signals connected to package pins or balls. Active pins depend on the PHY type selected (MII or RMII) and on the device configuration. *Table 618* shows the internal Ethernet signals.

**Table 617. Ethernet peripheral pins**

<table>
<thead>
<tr>
<th>Port name</th>
<th>Digital port type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETH_COL</td>
<td>Input</td>
<td>Collision detection signal, MII only.</td>
</tr>
<tr>
<td>ETH_CRS</td>
<td>input</td>
<td>Carrier sense signal, MII only</td>
</tr>
<tr>
<td>ETH_REF_CLK</td>
<td>Input</td>
<td>RMII reference clock</td>
</tr>
<tr>
<td>ETH_RX_CLK</td>
<td>Input</td>
<td>MII timing reference for Rx data transfers</td>
</tr>
<tr>
<td>ETH_RXD[3:0]</td>
<td>Input</td>
<td>Receive data. 4 pins for MII, 2 for RMII.</td>
</tr>
<tr>
<td>ETH_RX_DV</td>
<td>Input</td>
<td>Receive data valid</td>
</tr>
<tr>
<td>ETH_CRS_DV</td>
<td>Input</td>
<td>RMII: Carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.</td>
</tr>
<tr>
<td>ETH_RX_ER</td>
<td>Input</td>
<td>Receive error</td>
</tr>
<tr>
<td>ETH_TX_CLK</td>
<td>Input</td>
<td>MII timing reference for Tx data transfers</td>
</tr>
<tr>
<td>ETH_TXD[3:0]</td>
<td>Output</td>
<td>Transmit data. 4 pins for MII, 2 for RMII.</td>
</tr>
<tr>
<td>ETH_TX_EN</td>
<td>Output</td>
<td>Transmit data enable</td>
</tr>
<tr>
<td>ETH_TX_ER</td>
<td>Output</td>
<td>Transmit error</td>
</tr>
<tr>
<td>ETH_MDC</td>
<td>Output</td>
<td>Management data clock</td>
</tr>
<tr>
<td>ETH_MDIO</td>
<td>Input/output</td>
<td>Management data</td>
</tr>
</tbody>
</table>
The Ethernet peripheral is composed of 4 main functional modules:

- **The control and status register module (CSR)** that controls the registers access through AHB 32-bit slave interface
- The direct memory access interface (DMA)
  This is the logical DMA module with one physical channel for reception and 1 for transmission. It controls the data transfers between MAC and system memory through the AMBA AHB 32-bit master interface.
- **The media access control module (MAC)** in charge of implementing the Ethernet protocol
- **The MAC transaction layer (MTL)** in charge of controlling the data flow between application and MAC.

A protocol adaption module is added to support the RMII PHY Media Independent Interfaces.
57.4.1 DMA controller

The DMA has independent Transmit (Tx) and Receive (Rx) engines. The Tx engine transfers data from the system memory to the MAC Transaction Layer (MTL), whereas the Rx engine transfers data from the device port (PHY) to the system memory.

The controller uses descriptors to efficiently move data from source to destination with minimal application CPU intervention. The DMA is designed for packet-oriented data transfers such as packets in Ethernet. The controller can be programmed to interrupt the application CPU for situations such as Packet Transmit and Receive Transfer completion, and other normal or error conditions.

DMA data structures

The DMA and the application communicate through the following two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

The DMA transfers the data packets received by the MAC to the Rx buffer in system memory and Tx data packets from the Tx buffer in the system memory. The descriptors that reside in the system memory contain the pointers to these buffers.

The base address of each list is written to the respective Tx and Rx registers: Channel Tx descriptor list address register (ETH_DMACTXDLAR) and Channel Rx descriptor list address register (ETH_DMACRXDLAR).

---

1. For a definition of the internal signals, refer to Table 618.
2. Refer to RCC chapter "Clock distribution for Ethernet" for a detailed description of the Ethernet clock architecture.
The descriptor list is forward linked and the next descriptor is always considered at a fixed offset to the current one. The number of descriptors in the list is programmed in the respective Tx/Rx, Channel Tx descriptor ring length register (ETH_DMACTXRLR) and Channel Rx descriptor ring length register (ETH_DMACRXRLR).

Once the DMA processes the last descriptor in the list, it automatically jumps back to the descriptor in the List address register to create a descriptor ring. The descriptor lists reside in the physical memory address space of the application. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used and physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the application physical memory space and consists of an entire packet or part of a packet, but cannot exceed a single packet. Buffers contain only data. The buffer status is saved in the descriptor. Data chaining refers to packets that span multiple data buffers. However, a single descriptor cannot span multiple packets. The DMA skips to the data buffer of next packet when EOP is detected.

Descriptors are specified in Section 57.10: Descriptors.

DMA arbitration

The DMA module incorporates an arbiter that performs the arbitration between the Tx and Rx channels accesses from the AHB master interface. The following two types of arbitrations are supported and can be selected through DMA mode register (ETH_DMAMR):

- Round-robin arbitration: the arbiter allocates the data bus between Rx and Tx in ratio set by Bits [14:12] of ETH_DMAMR.
- Fixed-priority arbitration: by default Rx DMA always gets priority over Tx DMA for data access. Setting bit 11 of ETH_DMAMR register gives priority to the Tx DMA.
DMA transmission in default mode

The Tx DMA engine in default mode proceeds as follows:

1. The application sets up the Transmit descriptor (TDES0–TDES3) and sets the Own bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet Packet data.

2. The application shifts the Descriptor tail pointer offset value of the Transmit channel.

3. The DMA fetches the descriptor from the application memory.

4. If the DMA detects one of the following conditions, the transmission from that channel is suspended, bit 2 and 16 of the corresponding DMA channel Status register are set, and the Tx engine proceeds to step 11:
   - The descriptor is flagged as owned by the application (TDES3[31] = 0).
   - The descriptor tail pointer is equal to the current descriptor pointer in Ring Descriptor list mode.
   - An error condition occurs.

5. If the acquired descriptor is flagged as owned by the DMA (TDES3[31] = 1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.

6. The DMA fetches the Transmit data from the system memory and transfers the data to the MTL for transmission.

7. If an Ethernet packet is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3 through 7 are repeated until the end-of-Ethernet-packet data is transferred to the MTL.

8. When packet transmission is complete, if IEEE 1588 timestamp feature was enabled for the packet (as indicated in the Tx status), the timestamp value obtained from MTL is written to the Tx descriptor (TDES0 and TDES1) that contains the EOP buffer. The status information is written to this Tx descriptor (TDES3). The application now owns this descriptor because the Own bit is cleared during this step. If the timestamp feature is disabled for this packet, the DMA does not alter TDES0 and TDES1 contents.

9. Bit 0 of Channel status register (ETH_DMACSR) is set after completing transmission of a packet that has Interrupt on Completion (TDES2[31]) set in its Last Descriptor. The DMA engine returns to step 3.

10. In the Suspend state, the DMA tries to acquire the descriptor again (and thereby return to step 3). A poll demand command is triggered by writing any value to the Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR) when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared. If the application stopped the DMA by clearing Bit 0 of Transmit control register of corresponding DMA channel, the DMA enters the Stop state.
**DMA transmission in OSP (Operate on Second Packet) mode**

In Run state, if bit 4 is set in the Channel transmit control register (ETH_DMACTXCR), the Transmit process can simultaneously acquire two packets without closing the Status descriptor of the first packet. While the Transmit process completes the first packet transfer, it immediately polls the Transmit descriptor list for the second packet. If the second packet is valid, the Transmit process transfers this packet before writing the status information of the first packet.
In OSP mode, DMA transmission in the Run state operates as described in the following sequence:

1. The DMA executes steps 1 to 7 of the DMA transmission sequence in default mode (see Section: DMA transmission in default mode).
2. The DMA fetches the next descriptor without closing previous packet last descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and jumps to step 7.
4. The DMA fetches the Transmit packet from the system memory and transfers the packet to the MTL until the EOP data is transferred, closing the intermediate descriptors if this packet is split across multiple descriptors.
5. The DMA waits for the packet transmission status and timestamp of previous packet. When the status is available, the DMA writes the timestamp to TDES0 and TDES1 if such timestamp was captured (as indicated by a status bit). The DMA writes the status, with a cleared Own bit, to the corresponding TDES3, thus closing the descriptor. If Timestamp feature is not enabled for the previous packet, the DMA does not alter the contents of TDES2 and TDES3.
6. The Transmit interrupt is set (if enabled). The DMA fetches the next descriptor and proceeds to step 3 (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (step 7).
7. In Suspend mode, if a pending status and timestamp are received from the MTL, the DMA performs the following operations:
   a) The DMA writes the timestamp (if enabled for the current packet) to TDES2 and TDES3.
   a) The DMA writes the status to the corresponding TDES3.
   a) The DMA sets the relevant interrupts and returns to Suspend mode.
   If no status is pending and the application stopped the DMA by clearing bit 0 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state.
8. The DMA can exit Suspend mode and enter the Run state (it goes either to step 1 or to step 2 depending on pending status) only after receiving a Transmit Poll demand in Transmit Descriptor Tail Pointer register of corresponding channel.

A description of the basic DMA transmission flow in OSP mode in given in Figure 796: Receive DMA flow.
Figure 795. DMA transmission flow (OSP mode)

- Start Tx DMA
- (Re-)fetch next descriptor from Tx Queue
  - Own bit set?
    - Yes
    - No, Close intermediate descriptor
  - No
  - Packet read completely?
    - Yes
    - Second packet?
      - Yes
      - No
      - No
    - No
  - Yes
  - Timestamp present?
    - Yes
    - Write timestamp to TDES0 and TDES1
    - No
    - Wait status word to TDES3
  - No
  - End of descriptor ring?
    - Yes
    - Tx DMA queue suspended
    - No
  - Prev. packet status available?
    - Yes
    - Tx Poll demand?
      - Yes
      - No
      - No
    - No
    - DMA Tx stopped?
      - Yes
      - Stop DMA
      - No
**DMA reception**

In the Receive path, the DMA reads a packet from the MTL receive queue and writes it to the packet data buffers of the corresponding DMA channel.

The reception sequence for Rx DMA engine is as follows (see also Figure 796: Receive DMA flow):

1. The application sets up the Rx descriptors (RDES0-RDES3) and the Own bit (RDES3[31]). The application must set the correct value in the Receive descriptor tail pointer register of corresponding DMA channel.

2. When bit 0 of `Channel receive control register (ETH_DMACRXCR)` is set, the DMA enters the Run state. The DMA looks for free descriptors based on the Rx Current Descriptor and Descriptor tail pointer register values. If there are no free descriptors, the DMA channel enters the Suspend state and goes to step 11.

3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from acquired descriptors.

4. If IEEE 1588 timestamping is enabled and the timestamp is available for the previous packet, the DMA writes the timestamp (if available) to the RDES0 and RDES1 of current descriptor and sets the CTXT field (RDES3[30]).

5. The DMA processes the incoming packets and stores them in the data buffers of acquired descriptor.

6. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to step 10.

7. The DMA retrieves the status of the Receive frame from the MTL and writes the status word to current descriptor with the Own bit cleared and the Last descriptor bit set.

8. The DMA writes the Frame Length to RDES3 and the VLAN tag to RDES0. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor.

9. The DMA stores the timestamp (if available). The DMA writes the context descriptor after the last descriptor for the current packet (in the next available descriptor).

10. If more descriptors are available in the Rx DMA descriptor ring, go to step 3, otherwise go to the Suspend state (step 11).

11. The Receive DMA exits the Suspend state when a Receive Poll demand is given and the application increments the channel Receive tail pointer register. The engine proceeds to step 2 and fetches again the next descriptor.
Figure 796. Receive DMA flow

- **Start Rx DMA**
- **Stop Rx DMA**
- **(Re)Fetch next descriptor**
- **Rx DMA stopped?**
  - Yes: **Supsend Rx DMA**
  - No: **Receive Poll demand**
- **Packet data available?**
  - Yes: **Write data to buffer**
  - No: **Wait for packet data**
- **Close RDES3 as intermediate descriptor**
  - Yes: **Set pending timestamp**
  - No: **Packet transfer complete?**
    - Yes: **Write timestamp to RDES0 and RDES1**
    - No: **End of descriptor ring?**
      - Yes: **Close RDES3 as last descriptor**
      - No: **Packet data available?**
        - Yes: **Write data to buffer**
        - No: **Wait for packet data**
- **Timestamp pending?**
  - Yes: **Start Rx DMA**
  - No: **Close RDES3 as last descriptor**
- **Own bit set?**
  - Yes: **Stop Rx DMA**
  - No: **Timestamp present?**
    - Yes: **Set pending timestamp**
    - No: **Close RDES3 as intermediate descriptor**
- **Rx DMA stopped?**
  - Yes: **Stop Rx DMA**
  - No: **End of descriptor ring?**
    - Yes: **Stop Rx DMA**
    - No: **Timestamp present?**
      - Yes: **Set pending timestamp**
      - No: **Close RDES3 as intermediate descriptor**
- **Supsend Rx DMA**
- **Clear pending timestamp**
Priority scheme for Tx DMA and Rx DMA

The DMA arbiter performs the arbitration between the Tx and Rx paths of DMA channel 0 to access descriptors and data buffers. The DMA arbiter supports two types of arbitration: fixed priority and weighted round-robin. The DA bit of the DMA mode register (ETH_DMAMR) specifies the arbitration scheme (fixed or weighted round-robin) between the Tx and Rx DMA of a given channel.

If the Tx DMA and Rx DMA of a given channel are enabled, the DMA which gets the bus when the channel gets control of the bus must be specified. The priority between the corresponding Tx DMA and Rx DMA can be configured through the TXPR field of the DMA mode register (ETH_DMAMR). For round-robin arbitration, the weighted priority between the Tx DMA and Rx DMA is configured through the PR field of the DMA mode register (ETH_DMAMR). Table 619 provides information about the priority scheme between Tx DMA and Rx DMA.

### Table 619. Priority scheme for Tx DMA and Rx DMA

<table>
<thead>
<tr>
<th>DMA mode register (ETH_DMAMR)</th>
<th>Priority scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR[2:0] TXPR DA</td>
<td></td>
</tr>
<tr>
<td>x x x 0 1</td>
<td>Rx always has priority over Tx</td>
</tr>
<tr>
<td>0 0 0 0 0</td>
<td>Tx and Rx have equal priority. Rx gets the access first on simultaneous requests.</td>
</tr>
<tr>
<td>0 0 1 0 0</td>
<td>Rx has priority over Tx in ratio 2:1.</td>
</tr>
<tr>
<td>0 1 0 0 0</td>
<td>Rx has priority over Tx in ratio 3:1.</td>
</tr>
<tr>
<td>0 1 1 0 0</td>
<td>Rx has priority over Tx in ratio 4:1.</td>
</tr>
<tr>
<td>1 0 0 0 0</td>
<td>Rx has priority over Tx in ratio 5:1.</td>
</tr>
<tr>
<td>1 1 0 0 0</td>
<td>Rx has priority over Tx in ratio 6:1.</td>
</tr>
<tr>
<td>1 1 1 0 0</td>
<td>Rx has priority over Tx in ratio 7:1.</td>
</tr>
<tr>
<td>x x x 1 1</td>
<td>Tx always has priority over Rx</td>
</tr>
<tr>
<td>0 0 0 1 0</td>
<td>Tx and Rx have equal priority. Tx gets the access first on simultaneous requests.</td>
</tr>
<tr>
<td>0 0 1 1 0</td>
<td>Tx has priority over Rx in ratio 2:1.</td>
</tr>
<tr>
<td>0 1 0 1 0</td>
<td>Tx has priority over Rx in ratio 3:1.</td>
</tr>
<tr>
<td>0 1 1 1 0</td>
<td>Tx has priority over Rx in ratio 4:1.</td>
</tr>
<tr>
<td>1 0 0 1 0</td>
<td>Tx has priority over Rx in ratio 5:1.</td>
</tr>
<tr>
<td>1 0 1 1 0</td>
<td>Tx has priority over Rx in ratio 6:1.</td>
</tr>
<tr>
<td>1 1 0 1 0</td>
<td>Tx has priority over Rx in ratio 7:1.</td>
</tr>
<tr>
<td>1 1 1 1 0</td>
<td>Tx has priority over Rx in ratio 8:1.</td>
</tr>
</tbody>
</table>
57.4.2 MTL

The MAC Transaction Layer (MTL) provides the FIFO memory interface to buffer and regulate the packets between the application system memory and the MAC. It also enables the data to be transferred between the application clock and MAC clock domains. The MTL layer features two 32-bit wide data paths: the Transmit path and the Receive Path.

- Transmit path
  The application or internal DMA pushes the Ethernet packets read from the application or system memory into the Tx FIFO. The packet is then popped out and transferred to the MAC when the queue threshold is reached (threshold mode) or complete packet is in the queue (store-and-forward mode). When EOP is transferred, the status of the transmission is taken from the MAC and transferred back to the application or internal DMA. The Tx queue size is 2048 bytes.

- Receive path
  The MTL Rx module receives the packets from the MAC and pushes them into the Rx queue. The status (fill level) of the queue is indicated to the application or to DMA when it crosses the configured Receive threshold (RTC bits[1:0] defined in Rx queue operating mode register (ETH_MTLRXQOMR)), or when the complete packet was received. The MTL also indicates the queue fill level so that the DMA can initiate preconfigured burst transfers towards the master interface. The Rx queue size is 2048 bytes.

57.4.3 MAC

The MAC is responsible of the Ethernet protocol processing. In Transmission mode, it receives data from MTL before transferring it to the PHY interface. In Reception mode, the MAC receives data from the PHY interface before transferring them to the Rx FIFO of the MTL module.

This section briefly describes transmission and reception sequences.

MAC transmission

The transmission sequence is as follows:

1. Transmission is initiated when the MTL application pushes in data with the SOP (Start of packet) signal asserted.
2. When the SOP signal is detected, the MAC accepts the data and begins the transmission to the MII.
3. When the EOP (End of packet) is transferred to the MAC, the MAC does one of the following:
   - The MAC completes the normal transmission and provides the transmission status to the MTL.
   - If a normal collision (in Half-duplex mode) occurs during transmission, the MAC provides the Transmit status to the MTL, with the Retry bit set. The MAC provides the Retry request till one of the following is true:
     - the packet was successfully transmitted;
     - the maximum number of Retry requests expires. In this case, the MAC aborts the packet transmission with Excessive Collision Transmit status. The MAC accepts and drops all further data until the next SOP is received. The MTL block should retransmit the same packet from SOP when a Retry request (in the Status) is observed from the MAC.
– If any one of the following event happens, the MAC aborts the packet transmission:
  no carrier (Half-duplex mode)
  loss of carrier (Half-duplex mode)
  excessive deferral (Half-duplex mode)
  late collisions (Half-duplex mode)
  jabber

the MAC accepts and drops all further data until the next SOP is received.

4. The MAC issues an underflow status if the MTL is not able to provide the data continuously during the transmission. The MAC accepts and drops all further data until the next SOP is received.

5. During the normal transfer of a packet from MTL, if the MAC receives a SOP without getting an EOP for the previous packet, it ignores the SOP and considers the new packet as continuation of the previous packet.

*Figure 797: Overview of MAC transmission flow* illustrates the MAC transmission process flow.
Figure 797. Overview of MAC transmission flow

1. Start
2. Wait for data & SOP From MTL
3. SOP asserted by MTL? YES
4. Wait for IPG/any back-off delay (hal-duplex)
5. Transmit preamble+SFD+Data received from the MTL to the PHY
6. No carrier/Carrier loss/Excessive deferral/Late collisions/Jabber? YES
   - Drop all the data received from MTL and abort transmission
7. NO
   - Collision
     - A. EOP asserted by MTL?
     - B. Underflow asserted by MAC?
     - Condition A
     - Condition B
   - Normal transmission completed and Transmission status conveyed to MTL
8. Send status to MTL with Retry bit set
9. Condition D: Retry_count ≤ retry_limit
10. YES
    - Drop all the data received from MTL and abort transmission
11. NO
MAC reception

A receive operation is initiated when the MAC detects an SFD on MII. The MAC strips the preamble and SFD before proceeding to process the packet. The header fields are checked for filtering and the FCS field used to verify the CRC for the packet. The received packet is stored in a shallow buffer until the address filtering is performed. The packet is dropped in the MAC if it fails the address filter.

The reception sequence is as follows:

1. When the receive data valid signal (RxDV) of MII becomes active, the Receive State Machine (RSM) starts looking for the SFD field (0xD nibble).
   The state machine drops received packets until it detects SFD.

2. When SFD is detected, the state machine starts sending the data of Ethernet packet to the RPC module, beginning with the first byte following the SFD (destination address).

3. If IEEE 1588 timestamp feature is enabled, the MAC takes a snapshot of the system time at which SFD of any packet is detected on MII. If this packet is not dropped during MAC filtering, the timestamp is passed to the application. The MAC converts the received nibble data into bytes and forwards the valid packet data to the RFC module.

4. The Receive State Machine decodes the Length/Type field of the Ethernet packet being received.
   If the Length/Type field is less than 1,536 and if the MAC is programmed for the Auto CRC/Pad Stripping (bit 20 of the Operating mode configuration register (ETH_MACCR)), the state machine sends the packet data up to the count specified in the Length/Type field and starts dropping bytes (including the FCS field). The state machine decodes the Length/Type field and checks for the Length interpretation.

5. If the Length/Type field is greater than or equal to 1,536, the RPE module sends all received Ethernet packet data to the RFC module if you have not enabled the CRC stripping for Type packet in Bit 21 of the Operating mode configuration register (ETH_MACCR). However, if the CRC stripping has been enabled for Type packets and not enabled the Receive Checksum Offload Engine, the MAC strips and drops the last 4 bytes of all packets of ether type before forwarding the packets to the application.

6. By default, the MAC is programmed for watchdog timer to be enabled, that is, packets above 2,048 (10,240 if Jumbo Packet is enabled) bytes (DA + SA + LT + DATA + PAD + FCS) are cut off at the RPE module. In addition, you can use a programmable watchdog timer (bit 16 of Watchdog timeout register (ETH_MACWTR)) to override the fixed timeout of 2,048 or 10,240 bytes. You can disable the watchdog timer by programming bit 19 of Operating mode configuration register (ETH_MACCR). However, even if the watchdog timer is disabled, a packet greater than 32 Kbytes is cut off and a watchdog timeout status is given.
Figure 798. MAC reception flow

1. Start
2. Wait for phy_rxRxdv_i from GMII/MII
3. Wait until SFD is detected
4. Start sending the received Ethernet packet to the application layer, starting from DA field
5. IEEE 1588 timestamp feature enabled?
   - YES: Store the snapshot of the system time when SFD detected
   - NO: Length/Type field < 1536 & Auto CRC/Pad stripping enabled?
      - NO: Length/Type field ≥ 1536
      - YES: CRC stripping for Type packets enabled?
         - YES: Receive checksum
         - NO: CRC error? OR Packet to be filtered?
            - NO: CRC error? OR Packet to be filtered?
            - YES: Drop the packet and send the Status to the Application layer
            - NO: Send the packet, timestamp and Status to the Application layer
   - NO: Send only the number of bytes (specified in the Length field) of the received Ethernet packet to RFC. Drop extra padding and FCS field
      - CRC error? OR Packet to be filtered?
        - NO: CRC error? OR Packet to be filtered?
        - YES: Drop the packet and send the Status to the Application layer
        - NO: Send all received Ethernet packet data to RFC
          - CRC error? OR Packet to be filtered?
            - NO: CRC error? OR Packet to be filtered?
            - YES: Drop the last 4 bytes of all Ether type packets and send the remaining bytes to RFC
            - NO: Drop the packet and send the Status to the Application layer

57.5 **Ethernet functional description: MAC**

57.5.1 **Double VLAN processing**

The Ethernet peripheral supports the double VLAN (Virtual LAN) tagging feature in which the MAC can process up to two VLAN tags (inner and outer).

The MAC supports the following:
- Insertion, replacement, or deletion of up to two VLAN tags in the Transmit path
- Packet filtering and stripping based on any one of the two VLAN Tags in the Receive path. Stripping and providing up to two VLAN Tags in the Receive path as a part of the Receive status

**Transmit path**

*Table 620: Double VLAN processing features in Tx path* describes the features supported by the MAC on the Transmit side.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for C-VLAN and S-VLAN Tag types</td>
<td>The inner or outer VLAN tag can be of C-VLAN and S-VLAN type. The VLAN type is specified through the CSVL bit of VLAN inclusion register (ETH_MACVIR) and Inner VLAN inclusion register (ETH_MACIVIR), respectively. The Ethernet peripheral supports processing of any sequence of outer and inner VLAN tags. However, it does not support the C-VLAN S-VLAN sequence. The MAC does not check whether the packet provided by the application has a valid sequence of the VLAN Tag types or the insertion or replacement operation results in invalid sequence of VLAN Tag type. Therefore, the application must provide correct sequence of VLAN Tag types and program the MAC in such a way that it results in correct sequence of VLAN Tag types in the transmitted packet. The application must ensure the following: – The inner tag should not be S-VLAN when outer C-VLAN Tag insertion is enabled. – The outer tag should not be C-VLAN when inner S-VLAN Tag insertion is enabled. – The inner tag should not be S-VLAN when outer tag should be replaced with C-VLAN. – The outer tag should not be C-VLAN when inner tag should be replaced with S-VLAN.</td>
</tr>
<tr>
<td>VLAN Tag deletion</td>
<td>VLAN tag deletion can be enabled for outer or inner tag through VLC field in the VLAN inclusion register (ETH_MACVIR) or Inner VLAN inclusion register (ETH_MACIVIR), respectively. When VLAN deletion is enabled, the MAC deletes the tag present at the corresponding position. When a packet has only one tag, it is considered as the outer tag. If inner tag deletion is enabled and the packet has only one tag, the MAC does not delete the tag.</td>
</tr>
<tr>
<td>VLAN Tag Insertion or Replacement</td>
<td>VLAN tag insertion or replacement can be enabled for outer or inner tag through VLC field in the VLAN inclusion register (ETH_MACVIR) or Inner VLAN inclusion register (ETH_MACIVIR), respectively. When VLAN tag insertion or replacement is enabled, the VLTI bit in the previous register is used to determine whether the VLAN tag should be taken from the register or the control word.</td>
</tr>
</tbody>
</table>
Receive path

Table 621: Double VLAN processing in Rx path describes the features supported by the MAC on the Receive side and the corresponding bits in the VLAN tag register (ETH_MACVTR).

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outer or inner VLAN tag-based filtering</td>
<td>The MAC can filter packets based on the outer or inner VLAN tag through the ERIVLT bit.</td>
</tr>
<tr>
<td>C-VLAN or S-VLAN tag-based filtering</td>
<td>The MAC can filter packets based on the C-VLAN or S-VLAN type based on the ERSVLM bit.</td>
</tr>
<tr>
<td>Outer and Inner VLAN Tag stripping</td>
<td>The MAC can strip the outer and inner VLAN Tags from received frame based on the EVLS and EIVLS bits.</td>
</tr>
<tr>
<td>16-bit outer and inner VLAN Tag and Type in Rx status</td>
<td>The MAC can provide the 16-bit outer and inner VLAN Tag and Type in the Rx status based on the EVLRXS and EIVLRXS bits, respectively.</td>
</tr>
<tr>
<td>Disabling or skipping checking of outer VLAN Tag type</td>
<td>The MAC can disable or skip checking of outer VLAN Tag type to match C-VLAN or S-VLAN based on the DOVLTC bit.</td>
</tr>
</tbody>
</table>

57.5.2 Source address and VLAN insertion, replacement, or deletion

Source address insertion or replacement

The software can use the SA (source address) insertion or replacement feature to instruct the MAC to do the following for Tx packets:

- Insert the content of the MAC Address registers in the SA field
- Replace the content of the SA field with the content of the MAC Address registers

When SA insertion is enabled, the application must ensure that the packets sent to the MAC do not have the SA field. The MAC does not check whether the SA field is present in the Transmit packet and it inserts the content of MAC Address Registers in the SA field. Similarly, when SA replacement is enabled, the application must ensure that the SA field is present in the packets sent to the MAC. The MAC replaces the six bytes following the Destination Address field in the Transmit packet with the content of the MAC Address Registers.

SA insertion or replacement feature can be enabled for all Transmit packets or selective packets:

- Enabling SA insertion or replacement for all packets
  To enable this feature for all packets, program the SARC field of the Operating mode configuration register (ETH_MACCR).
- Enabling SA insertion or replacement for selective packets
  To enable this feature for selective packets, use the following program the SA Insertion Control field (bits[25:23] of Transmit Descriptor Word 3/TDES3, refer to Section 57.10.3: Transmit descriptor) in the first Transmit descriptor of the packet. When Bit 25 of TDES3 is set, the SA Insertion Control field indicates insertion or
replacement by MAC Address1 registers. When bit 25 of TDES3 is reset, it indicates insertion or replacement by MAC Address 0 registers.

If MAC Address1 registers are not enabled, the MAC Address0 registers are used for insertion or replacement whatever of the value of the most-significant bit of the SA Insertion Control field.

**VLAN insertion, replacement, or deletion**

The software can use the VLAN insertion, replacement, or deletion feature to instruct the MAC to do the following for Tx packets:

- Delete the VLAN Type and VLAN Tag fields
- Insert or replace the VLAN Type and VLAN Tag fields

Insertion or replacement is performed based on the setting of VLTI bit in the **VLAN inclusion register (ETH_MACVIR)** as described in **Table 622: VLAN insertion or replacement based on VLTI bit**.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLTI bit is set</td>
<td>The MAC inserts or replaces the following:</td>
</tr>
<tr>
<td></td>
<td>VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of VLAN inclusion register (ETH_MACVIR))</td>
</tr>
<tr>
<td></td>
<td>VLAN Tag field with VT field of Transmit context descriptor of the packet</td>
</tr>
<tr>
<td>VLTI bit is reset</td>
<td>The MAC inserts or replaces the following:</td>
</tr>
<tr>
<td></td>
<td>VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of VLAN inclusion register (ETH_MACVIR))</td>
</tr>
<tr>
<td></td>
<td>VLAN Tag field with the VLT field of VLAN inclusion register (ETH_MACVIR)</td>
</tr>
</tbody>
</table>

When VLAN replacement or deletion is enabled, the MAC checks if the VLAN Type field (0x8100 or 0x88A8) is present after the DA and SA fields in the Transmit packet. The replace or delete operation does not occur if the VLAN Type field is not detected in two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check the presence of VLAN Type field in the Transmit packet and just inserts the VLAN Type and VLAN Tag fields.

You can enable the VLAN insertion, replacement, or deletion feature for all Tx packets or selective packets:

- To enable this feature for all packets, program the VLC and VLP fields of **VLAN inclusion register (ETH_MACVIR)**.
- To enable this feature for selective packets, program the VTIR field of TDES2 Normal Descriptor (see **Table 655: TDES2 normal descriptor (read format)**).

In addition, the VLP (VLAN Priority control) bit must be reset in **VLAN inclusion register (ETH_MACVIR)** (for outer VLAN) and **Inner VLAN inclusion register (ETH_MACIVIR)** (in inner VLAN) for the MAC to take the control inputs from the host, depending on the configuration.
57.5.3 Packet filtering

The MAC supports the following types of filtering for Rx packets:

- **MAC source or destination address filtering**: the Address Filtering Module (AFM) checks the source address and destination address fields of each incoming packet.
- **VLAN filtering**: the MAC supports the VLAN tag-based and VLAN Hash filtering.
- **Layer 3 and Layer 4 filtering**: Layer 3 filtering refers to IP source address and destination address filtering. Layer 4 filtering refers to source port and destination port filtering.

The three filter types can be cascaded. *Figure 799* shows the filtering sequence for Rx packets.

*Figure 799. Packet filtering sequence*

The sequence shown in *Figure 799* is valid when all the filters (L2, VLAN, L3, L4) are active. If any of the Layer filters are not enabled, that filter is bypassed and the subsequent filter is applied. A packet that fails any of the filters is discarded. However, the discarded packet can be forwarded to the host based on the register control.

For example, when RA bit of *Packet filtering control register (ETH_MACPFR)* is set to 1, all the discarded packets are forwarded to the host but with their packet status indicating the
specific filter failure. If RA bit is cleared to 0, VTFE and IPFE bits of Packet filtering control register (ETH_MACPFR) control if the packets that fail the VLAN filter and Layer 3-4 filter should be discarded or forwarded to the host.

**MAC source or destination address filtering**

The MAC address filtering module checks the source address (SA) and destination address (DA) fields of each incoming packet.

**Unicast destination address filtering**

The MAC supports 4 MAC addresses for unicast perfect filtering. If perfect filtering is selected (HUC bit of Packet filtering control register (ETH_MACPFR) is reset), the MAC compares all 48 bits of received unicast address with the programmed MAC address for any match. The default MacAddr0 is always enabled.

The MacAddr1 to MacAddr3 addresses are selected with an individual enable bit. You can mask each byte during comparison with corresponding received DA byte by setting the corresponding Mask Byte Control bit in MAC Address x high register (ETH_MACAxHR). This enables group address filtering for the DA.

In Hash filtering mode (when HUC bit is set), the MAC performs imperfect filtering for unicast addresses using a 64-bit Hash table. For Hash filtering, the MAC uses the upper 6 bits CRC of the received destination address to index the content of the Hash table. A value of 00000 selects bit 0 of selected register, and a value of 11111 selects bit 63 of Hash Table register. If the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast packet is considered to have passed the Hash filter; otherwise, the packet is considered to have failed the Hash filter.

**Multicast destination address filtering**

To program the MAC to pass all multicast packets, set the PM bit in Packet filtering control register (ETH_MACPFR). If the PM bit is reset, the MAC performs the filtering for multicast addresses based on the HMC bit of the Packet filtering control register (ETH_MACPFR).

In Perfect filtering mode, the multicast address is compared with the programmed MAC destination address registers. Group address filtering is also supported.

In Hash filtering mode, the MAC performs imperfect filtering using a 64-bit Hash table. The MAC uses the upper 6-bits CRC of received multicast address to index the content of the Hash table. A value of 00000 selects bit 0 of selected register and a value of 11111 selects bit 63 of the Hash Table register. If the corresponding bit is set to 1, the multicast packet is considered to have passed the Hash filter. Otherwise, the packet is considered to have failed the Hash filter.

**Hash or Perfect address filtering**

To configure the DA filter to pass a packet when its DA matches either the Hash filter or the Perfect filter, set the HPF bit and the corresponding HUC or HMC bits in Packet filtering control register (ETH_MACPFR). This is applicable to both unicast and multicast packets. If the HPF bit is reset, only one of the filters (Hash or Perfect) is applied to receive packet.

**Broadcast address filtering**

The MAC does not filter any broadcast packets by default. To program the MAC to reject all broadcast packets, set the DBF bit in Packet filtering control register (ETH_MACPFR).

**Unicast source address filtering**
The MAC can perform perfect filtering based on the source address field of received packets. By default, the MAC compares the SA field with the values programmed in the SA registers. You can configure the MAC Address registers to use SA instead of DA for comparison by setting bit 30 of MAC Address x high register (ETH_MACAxHR).

The MAC also supports group filtering with SA. You can filter a group of addresses by masking one or more bytes of the address. The MAC drops the packets that fail the SA filter if the SAF bit is set in Packet filtering control register (ETH_MACPFR). Otherwise, the result of the SA filter is given as a status bit in the Receive Status word (see Table 624). When the SAF bit is set, the SA filter and DA filter result is ANDed to decide whether the packet needs to be forwarded. This means that the packet is dropped if either filter fails. The packet is forwarded to the application only if the packet passes both filters in-order.

**Inverse filtering**

For DA and SA filtering, you can invert the filter-match result at the final output by setting the DAIF and SAIF bits of Packet filtering control register (ETH_MACPFR). The DAIF bit is applicable for both Unicast and Multicast DA packets. The result of the unicast or multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

Table 623 and Table 624 summarize the DA and SA filtering based on the type of packets received.

**Note:** When the RA bit of Packet filtering control register (ETH_MACPFR) is set, all packets are forwarded to the system along with the correct result of the address filtering in the Rx status.

### Table 623. Destination address filtering

<table>
<thead>
<tr>
<th>Packet type</th>
<th>PR</th>
<th>HPF</th>
<th>HUC</th>
<th>DAIF</th>
<th>HMC</th>
<th>PM</th>
<th>DBF</th>
<th>DA filter operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcast</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>Fail</td>
</tr>
<tr>
<td>Unicast</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Pass all packets</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Pass on Perfect/Group filter match</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Fail on Perfect/Group filter match</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Pass on Hash filter match</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Fail on Hash filter match</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Pass on Hash or Perfect/Group filter match</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Fail on Hash or Perfect/Group filter match</td>
</tr>
<tr>
<td>Multicast</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Pass all packets</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>Pass all packets</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>Pass on Perfect/Group filter match and drop Pause packets if PCF = 0x</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>Pass on Hash filter match and drop Pause packets if PCF = 0x</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>Pass on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x</td>
</tr>
</tbody>
</table>
VLAN filtering

The MAC supports Perfect and Hash VLAN filtering. Refer to Section 57.9.14: Programming guidelines to perform VLAN filtering on the receive for detailed programming steps.

VLAN tag Perfect filtering

In VLAN tag Perfect filtering, the MAC compares the VLAN tag of received packet and provides the VLAN packet status to the application. Based on the programmed mode, the MAC compares the lower 12 bits or all 16 bits of received VLAN tag to determine the perfect match.

If VLAN tag Perfect filtering is enabled, the MAC forwards the VLAN-tagged packets along with VLAN tag match status and drops the VLAN packets that do not match. You can also enable the inverse matching for VLAN packets by setting the VTIM bit of VLAN tag register (ETH_MACVTR). In addition, you can enable matching of S-VLAN tagged packets along with the default C-VLAN tagged packets by setting the ESVL bit of VLAN tag register (ETH_MACVTR). The VLAN packet status bit (bit 10 of RDES0) indicates the VLAN tag match status for the matched packets.

Note: The source or destination address (if enabled) has precedence over the VLAN tag filters. This means that a packet that fails the source or destination address filter is dropped irrespective of the VLAN tag filter results.

VLAN tag Hash filtering

The 16-bit VLAN Hash Table is used for group address filtering based on the VLAN tag. The VLAN tag Hash filtering feature can be enabled using the VTHM (VLAN tag Hash Table match enable) bit of the VLAN tag register (ETH_MACVTR). If the VTHM bit is set, the most significant four bits of CRC-32 of VLAN tag are used to index the content of the VLAN Hash Table register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the VLAN tag of the packet matched and the packet should be forwarded. A value of 0 indicates that VLAN-tagged packet should be dropped.

<table>
<thead>
<tr>
<th>Packet type</th>
<th>SAIF</th>
<th>SAF</th>
<th>SA filter operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unicast</td>
<td>1</td>
<td>X</td>
<td>Pass all packets.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>Pass status on Perfect or Group filter match but do not drop packets that fail</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>Fail status on Perfect or Group filter match but do not drop packet</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>Pass on Perfect or Group filter match and drop packets that fail</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>Fail on Perfect or Group filter match and drop packets that fail</td>
</tr>
</tbody>
</table>

### Table 623. Destination address filtering (continued)

<table>
<thead>
<tr>
<th>Packet type</th>
<th>PR</th>
<th>HPF</th>
<th>HUC</th>
<th>DAIF</th>
<th>HMC</th>
<th>PM</th>
<th>DBF</th>
<th>DA filter operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multicast</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>Fail on Perfect/Group filter match and drop Pause packets if PCF = 0x</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>Fail on Hash filter match and drop Pause packets if PCF = 0x</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>Fail on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x</td>
</tr>
</tbody>
</table>

### Table 624. Source address filtering

<table>
<thead>
<tr>
<th>Packet type</th>
<th>PR</th>
<th>SAIF</th>
<th>SAF</th>
<th>SA filter operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unicast</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Pass all packets.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Pass status on Perfect or Group filter match but do not drop packets that fail</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Fail status on Perfect or Group filter match but do not drop packet</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Pass on Perfect or Group filter match and drop packets that fail</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Fail on Perfect or Group filter match and drop packets that fail</td>
</tr>
</tbody>
</table>
Note: The 16 or 12 bits of VLAN Tag are considered for CRC-32 computation based on ETV bit in ETH_MACVTR register.

When ETV bit is reset, most significant four bits of CRC-32 of VLAN Tag are inverted and used to index the content of VLAN Hash table register (ETH_MACVHTR).

When ETV bit is set, most significant four bits of CRC-32 of VLAN Tag are directly used to index the content of VLAN tag register (ETH_MACVTR).

The MAC also supports the inverse matching for VLAN packets. In the inverse matching mode, when the VLAN tag of a packet matches the Perfect or Hash filter, the packet should be dropped. If the VLAN perfect and VLAN Hash match are enabled, a packet is considered as matched if either the VLAN Hash or the VLAN perfect filter matches. When inverse match is set, a packet is forwarded only when both perfect and Hash filters indicate mismatch.

Table 625 shows the different possibilities for VLAN matching and the final VLAN match status. When the RA bit of Packet filtering control register (ETH_MACPFR) is set, all packets are received and the VLAN match status is indicated in the VF bit of RDES2 normal descriptor (write-back format). When the RA bit is not set and the VTFE bit is set in Packet filtering control register (ETH_MACPFR), the packet is dropped if the final VLAN match status is Fail. In Table 625, value X means that this column can have any value.

When VLAN VID is programmed to 0 in the VL field of VLAN tag register (ETH_MACVTR), all VLAN-tagged packets are considered as perfect matched but the status of the VLAN Hash match depends on the VTHM and VTIM bits in VLAN tag register (ETH_MACVTR).

<table>
<thead>
<tr>
<th>VID</th>
<th>VLAN perfect filter match result</th>
<th>VTHM Bit</th>
<th>VLAN Hash filter match result</th>
<th>VTIM bit</th>
<th>Final VLAN match status</th>
</tr>
</thead>
<tbody>
<tr>
<td>VID = 0</td>
<td>Pass</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>Pass</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>Pass</td>
<td>1</td>
<td>Fail</td>
<td>1</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>Pass</td>
<td>1</td>
<td>Pass</td>
<td>1</td>
<td>Fail</td>
</tr>
<tr>
<td>VID!= 0</td>
<td>Pass</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>Fail</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>1</td>
<td>Fail</td>
<td>0</td>
<td>Fail</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>1</td>
<td>Pass</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>Pass</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>Fail</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>1</td>
<td>Pass</td>
<td>1</td>
<td>Fail</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>1</td>
<td>Fail</td>
<td>1</td>
<td>Pass</td>
</tr>
</tbody>
</table>

1. In this table, 'X' represents any value.
Layer 3 and Layer 4 filtering

The MAC supports Layer 3 and Layer 4 based packet filtering. The Layer 3 filtering refers to the IP Source or Destination Address filtering in the IPv4 or IPv6 packets whereas Layer 4 filtering refers to the Source or Destination Port number filtering in TCP or UDP.

The Layer 3 and Layer 4 packet filtering feature automatically enables the IPC Full Checksum Offload Engine on the Receive side. For Layer 3 or Layer 4 filtering operation, you must set the IPC bit of the Operating mode configuration register (ETH_MACCR) to enable the Rx Checksum Offload engine.

When Layer 3 and Layer 4 filtering is enabled, the packets are filtered in the following way:

- Matched packets
  The MAC forwards the packets that match all enabled fields to the application along with the status. The MAC gives the matched field status only if the IPC bit of the Operating mode configuration register (ETH_MACCR) is set and one of the following conditions is true:
    - All enabled Layer 3 and Layer 4 fields match.
    - At least one of the enabled field matches and other fields are bypassed or disabled

  When multiple Layer 3 and Layer 4 filters are enabled, any filter match is considered as a match. If more than one filter matches, the MAC provides the status of the lowest filter where Filter 0 is the lowest filter and Filter 3 is the highest filter. For example, if Filter 0 and Filter 1 match, the MAC gives the status corresponding to filter 0.

  **Note:** The source or destination address and VLAN tag filters (if enabled) have precedence over Layer 3 and Layer 4 filter. This means that a packet which fails the source or destination address or VLAN tag filter is dropped irrespective of the Layer 3 and Layer 4 filter results.

- Unmatched packets
  The MAC drops the packets that do not match any of the enabled fields. You can use the inverse match feature to block or drop a packet with specific TCP or UDP over IP fields and forward all other packets. The aborted or partial packets are dropped in the MTL Rx FIFO. If the Rx FIFO operates in the Threshold (cut-through) mode and the threshold is programmed to a small value. Such packet transfer to the application starts before the failed Layer 3 and Layer 4 filter results are available, the application may receive a partial packet with appropriate abort status.

- Non-TCP or UDP IP Packets
  By default, all non-TCP or UDP IP packets are bypassed from the Layer 3 and Layer 4 filters. You can optionally program the MAC to drop all non-TCP or UDP over IP packets.

Layer 3 filtering

The MAC supports perfect matching or inverse matching for IP Source Address and Destination Address. In addition, you can match the complete IP address or mask the lower bits matching, that is, compare all bits of the address except the specified lower mask bits.

For IPv6 packets filtering, you can enable the last four data registers of a register set to contain the 128-bit IP Source Address or IP Destination Address. The IP Source Address or Destination Address should be programmed in the order defined in the IPv6 specification, that is, the first byte of the IP Source Address or Destination Address in the received packet is in the higher byte of the register and the subsequent registers follow the same order.
For IPv4 packet filtering, you can enable the second and third data registers of a register set to contain the 32-bit IP Source Address and IP Destination Address. The remaining two data registers are reserved. The IP Source Address or Destination Address should be programmed in the order defined in the IPv4 specification, that is, the first byte of IP Source Address and Destination Address in the received packet in the higher byte of the respective register.

**Layer 4 filtering**

The MAC supports perfect matching or inverse matching for TCP or UDP Source and Destination Port numbers. However, you can program only one type (TCP or UDP) at a time. The first data register contains the 16-bit Source and Destination Port numbers of TCP or UDP, that is, the lower 16 bits for Source Port number and higher 16 bits for Destination Port number.

The TCP or UDP Source and Destination Port numbers should be programmed in the order defined in the TCP or UDP specification, that is, the first byte of TCP or UDP Source and Destination Port number in the received packet is in the higher byte of the register.

**Layer 3 and Layer 4 filters register set**

The MAC implements two sets of registers for Layer 3 and Layer 4 based packet filtering. In a register set, there is a control register, such as `L3 and L4 control 0 register (ETH_MACL3L4C0R)`, to control the packet filtering. In addition, there are five address registers to program the Layer 3 and Layer 4 fields to be matched, such as:

- **Layer4 Address filter 0 register (ETH_MACL4A0R)**
- **Layer3 Address 0 filter 0 register (ETH_MACL3A00R)**
- **Layer3 Address 1 filter 0 register (ETH_MACL3A10R)**
- **Layer3 Address 2 filter 0 register (ETH_MACL3A20R)**
- **Layer3 Address 3 filter 0 register (ETH_MACL3A30R)**

The second, and independent set of registers are: `L3 and L4 control 1 register (ETH_MACL3L4C1R)`, `Layer 4 address filter 1 register (ETH_MACL4A1R)`, `Layer3 address 0 filter 1 Register (ETH_MACL3A01R)`, `Layer3 address 1 filter 1 register (ETH_MACL3A11R)`, `Layer3 address 2 filter 1 Register (ETH_MACL3A21R)` and `Layer3 address 3 filter 1 register (ETH_MACL3A31R)`.

**57.5.4 IEEE 1588 timestamp support**

The IEEE 1588 standard defines a precision time protocol (PTP) which allows precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the submicrosecond range with minimal network and local clock computing resources.

This chapter contains the following sections:

- **IEEE 1588 timestamp support**
- **IEEE 1588 system time source**
- **IEEE 1588 auxiliary snapshots**
- **Flexible pulse-per-second output**
• **PTP timestamp offload function**
• **One-step timestamp**

### IEEE 1588 timestamp support

The Ethernet peripheral supports the IEEE 1588-2002 (version 1) and IEEE 1588-2008 (version 2). The IEEE 1588-2002 supports PTP transported over UDP/IP. The IEEE 1588 2008 supports PTP transported over Ethernet. The peripheral provides programmable support for both standards. It supports the following features:

- Support of both timestamp formats
- Optional snapshot of all packets or only PTP type packets
- Optional snapshot of only event messages
- Optional snapshot based on the clock type: ordinary, boundary, end-to-end transparent, and peer-to-peer transparent
- Optional selection of the node to act as master or slave for ordinary and boundary clock
- Identification of the PTP message type, version, and PTP payload in packets sent directly over Ethernet and sends the status
- Optional measurement subsecond time in digital or binary format

### Clock types

The MAC supports the following clock types defined in the IEEE 1588-2008 specifications:

- **Ordinary clock**

  The ordinary clock of a domain supports a single copy of the protocol. It has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecom applications, the ordinary clock can be associated with a timing demarcation device.

  The ordinary clock can be a grandmaster or a slave clock. It supports the following features:
  
  - Transmission and reception of PTP messages. The timestamp snapshot can be controlled as described in *Timestamp control Register (ETH_MACTSCR)*.
  - Maintenance of the data sets such as timestamp values.

  The table below shows the messages for which you can take the timestamp snapshot on the receive side for master and slave nodes.

  **Table 626. Ordinary clock: PTP messages for snapshot**

<table>
<thead>
<tr>
<th>Master</th>
<th>Slave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay_Req</td>
<td>SYNC</td>
</tr>
</tbody>
</table>

For an ordinary clock, you can take the snapshot of either of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the TSVER2ENA bit and selecting the snapshot mode in *Timestamp control Register (ETH_MACTSCR)*.

- **Boundary clock**

  The boundary clock typically has several physical ports which communicate with the network. The messages related to synchronization, master-slave hierarchy, and
signaling end in the protocol engine of the boundary clock. Such messages are not forwarded. The PTP message type status given by the MAC helps to identify the type of message and take appropriate action.

The boundary clock is similar to the ordinary clock except for the following features:

- The clock data sets are common to all ports of the boundary clock.
- The local clock is common to all ports of the boundary clock.

• End-to-end transparent clock
The end-to-end transparent clock supports the end-to-end delay measurement mechanism between the slave clocks and the master clock. The end-to-end transparent clock forwards all messages like normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the Ingress port to the Egress port.

The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet, inside the end-to-end transparent clock, is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both Ingress and Egress ports only for the messages mentioned in Table 627. You can take the snapshot by setting the SNAPTYPSEL bits to 10 in the Timestamp control Register (ETH_MACTSCR).

<table>
<thead>
<tr>
<th>PTP messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNC</td>
</tr>
<tr>
<td>Delay_Req</td>
</tr>
</tbody>
</table>

• Peer-to-peer transparent clock
In the peer-to-peer transparent clock, the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages with the link peer.

The peer-to-peer transparent clock differs from the end-to-end transparent clock in the way it corrects and handles the PTP timing messages. In all other aspects, it is identical to the end-to-end transparent clock.

The residence time of the Pdelay_Req and the associated Pdelay_Resp packets is added and inserted into the correction field of the associated Pdelay_Resp_Followup packet. Therefore, support for taking snapshot for the event messages related to Pdelay is added as shown in Table 635.
You can take the snapshot by setting the SNAPTYPESEL bit to 11 in Timestamp control Register (ETH_MACTSCR).

**Delay request-response mechanism**

The system or network is classified into the master and slave nodes for distributing the timing and clock information. Figure 800 shows the process that PTP uses for synchronizing a slave node with a master node by exchanging PTP messages.

**Figure 800. Networked time synchronization**

As shown in Figure 800, the PTP uses the following process:

1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the reference time information of the master. This message leaves the system of the master at \( t_1 \). This time must be captured for Ethernet ports at MII.
2. The slave receives the SYNC message and also captures the exact time, \( t_2 \), using its timing reference.
3. The master sends a Follow_up message to the slave, which contains \( t_1 \) information for later use.
4. The slave sends a Delay_Req message to the master and notes the exact time, \( t_3 \), at which this packet leaves the MII interface.

<table>
<thead>
<tr>
<th>PTP messages</th>
<th>Master clock time</th>
<th>Slave clock time</th>
<th>Data at slave clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNC</td>
<td>( t_1 )</td>
<td>( t_2 )</td>
<td>( t_1, t_2 )</td>
</tr>
<tr>
<td>Pdelay_Req</td>
<td>( t_2m )</td>
<td>( t_1, t_2 )</td>
<td>( t_1, t_2, t_3 )</td>
</tr>
<tr>
<td>Pdelay_Resp</td>
<td>( t_3m )</td>
<td>( t_3 )</td>
<td>( t_1, t_2, t_3, t_4 )</td>
</tr>
<tr>
<td></td>
<td>( t_4 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 628: Peer-to-peer transparent clock: PTP messages for snapshot
5. The master receives the message, capturing the exact time \( t_4 \), at which the message enters its system.
6. The master sends the \( t_4 \) information to the slave in the Delay Resp message.
7. The slave uses the four values of \( t_1, t_2, t_3, \) and \( t_4 \) to synchronize its local timing reference with the timing reference of the master.

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the MII interface. This timing information must be captured and returned to the software for proper implementation of PTP with high accuracy.

**Peer-to-peer PTP transparent clock (P2P TC) message support**

The IEEE 1588-2008 standard supports peer-to-peer PTP (Pdelay) message in addition to the Sync, Delay Request, Follow-up, and Delay Response messages. *Figure 801* shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

*Figure 801. Propagation delay calculation in clocks supporting peer-to-peer path correction*

As shown in *Figure 801*, the propagation delay is calculated as follows:
1. Port 1 issues a Pdelay_Req message and generates a timestamp (\( t_1 \)) for the Pdelay_Req message.
2. Port 2 receives the Pdelay_Req message and generates a timestamp (\( t_2 \)) for this message.
3. Port 2 returns a Pdelay_Resp message and generates a timestamp ($t_3$) for this message.

To minimize errors caused by frequency offset between the two ports, Port 2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. Port 2 returns any one of the following:

- Difference between the timestamps $t_2$ and $t_3$ in the Pdelay_Resp message
- Difference between the timestamps $t_2$ and $t_3$ in the Pdelay_Resp_Follow_Up message
- Timestamps $t_2$ and $t_3$ in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively

4. Port 1 generates a timestamp ($t_4$) on receiving the Pdelay_Resp message.

5. Port 1 uses all four timestamps to compute the mean link delay.

**Timestamp correction**

According to the IEEE 1588 specifications, a timestamp must be captured when the message timestamp point (leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet) crosses the boundary between the node and the network.

As the MAC takes the timestamp at an internal point far from the actual boundary of the node and network, this captured timestamp is corrected/updated for the ingress/egress path latency (including the delay in the PHY layers). Further correction is done for the inaccuracies/errors introduced due to the clock (MII Tx, Rx clock) being different at the capture point as compared to the PTP clock (clk_ptp_ref_i) that is used to generate the time. The resultant CDC (clock domain crossing) circuits add an error the depends on the clock period of the MII and PTP clocks.

**Ingress correction**

In the Receive side, the timestamp captured at the internal snapshot point is delayed (later in time) as compared to the time at which that packet SFD bit is received at the port boundary. Therefore, the captured timestamp must be reduced by the ingress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the Timestamp Ingress correction nanosecond register (ETH_MACTSICNR).

The correction value consists of the following three components:

1. External latency in the PHY layer between boundary point and the input of the core
   
   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has registers indicating the maximum and minimum ingress latency. The software can read these registers and determine the average ingress latency in the PHY. Alternatively (if the PHY does not support these registers), the ingress latency must be determined from the PHY datasheet or timing characteristics.

2. Internal latency from the input of the core to the internal capture point
   
   The latency differs based on the active PHY interface (such as MII or RMII) and the operating speed, as shown in Table 629.

3. CDC synchronization
   
   The CDC synchronization error is almost equal to twice the clock-period of the PTP clock (clk_ptp_ref_i).
The values determined from these three components should be added by the software and must be written into the TSIC field of the Timestamp Ingress correction nanosecond register (ETH_MACTSICNR).

Note: The value written to the register must be negative (two’s complement), because it has to be subtracted from the captured timestamp. The MAC receiver adds the value in this register to the captured timestamp and then gives the resultant value as the timestamp of the received packet.

When TSCTRLSSR bit in Timestamp control Register (ETH_MACTSCR) is set, the nanoseconds field of the captured timestamp is in decimal format with a granularity of 1 ns. So bit 31 of TSIC must be set to 1 (for negative value) and bits 30 to 0 must be programmed with \(10^{9} - \text{total ingress_correction_value[nanosecond part]}\) represented in binary. For example, if the required correction value is \(-5\) ns, then the value is 0xBB9A C9FB.

When TSCTRLSSR bit in Timestamp control Register (ETH_MACTSCR) is reset, the nanoseconds field of the captured timestamp is in binary format with a granularity of \(~0.466\) ns. Therefore, bits[30:0] must be written with \(2^{31} - \text{total ingress_correction_value}\) represented in binary with bit[31] = 1.

**Egress correction**

In the Transmit side, the timestamp captured at the internal snapshot point is earlier (advanced in time) as compared to the time at which that packet SFD bit is output at the port boundary. Therefore, the captured timestamp must be compensated by the egress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the Timestamp Egress correction nanosecond register (ETH_MACTSECNR).

The correction value consists of the following three components:

1. External latency in the PHY layer between the output of the core and the boundary of the port and the network
   
   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has registers indicating the maximum and minimum egress latency. The software can read these registers and determine the average egress latency in the PHY. Alternatively (if the PHY does not support these registers), the egress latency must be determined from the PHY datasheet or timing characteristics.

2. Internal latency from the internal capture point and the output of the core
   
   The latency differs based on the active PHY interface (RMII, MII, etc.) and the operating speed as shown in Table 629.

3. CDC synchronization error
   
   The timestamp correction because of synchronization is compensated by adding
   
   \[
   \text{EGRESS_SYNC_CORR} = (1 \times \text{PTP_CLK_PER} + 4 \times \text{TX_CLK_PER})
   \]

*Table 629* lists the Egress and Ingress latency values for various PHY interfaces:

<table>
<thead>
<tr>
<th>PHY interface</th>
<th>Egress latency</th>
<th>Ingress latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGMII 1 Gbps</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>RGMII 100 Mbps</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>RGMII 10 Mbps</td>
<td>400</td>
<td>400</td>
</tr>
</tbody>
</table>
Phy interface | Egress latency | Ingress latency
---|---|---
RMII 100 Mbps | 40 | 120
RMII 10 Mbps | 400 | 800

### Frequency range of reference timing clock

The timestamp information are transferred across asynchronous clock domains, from the MAC clock domain to the application clock domain. Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is four clock cycles of MII and three clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second packet.

The PTP clock frequency limitations are the following:

- **Maximum PTP clock frequency**
  - The maximum PTP clock frequency is limited by the maximum resolution of the reference time (10 ns at 100 MHz). In addition, the resolution or granularity of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.

- **Minimum PTP clock frequency**
  - The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes and the time taken for synchronizing the time with the MII clock domain. This relationship is given by the following equation:
    
    \[3 \times \text{PTP clock period} + 4 \times \text{MII clock period} \leq \text{Minimum gap between two SFDs}\]

  The MII clock frequency is fixed by IEEE specifications. Therefore, the minimum PTP clock frequency required for proper operation depends on the operating mode and operating speed of the MAC as shown in Table 630.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Minimum gap between two SFDs</th>
<th>Minimum PTP frequency with internal timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 Mbps Full-duplex</td>
<td>168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)</td>
<td>5 MHz</td>
</tr>
<tr>
<td>10 Mbps Half-duplex</td>
<td>48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)</td>
<td>5 MHz</td>
</tr>
<tr>
<td>100 Mbps full duplex</td>
<td>168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)</td>
<td>5 MHz</td>
</tr>
<tr>
<td>100 Mbps Half-duplex</td>
<td>48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)</td>
<td>5 MHz</td>
</tr>
</tbody>
</table>

### PTP processing and control
Table 631 shows the common message header for the PTP messages. This format is taken from the IEEE 1588-2008 specifications.

Table 631. Message format defined in IEEE 1588-2008

<table>
<thead>
<tr>
<th>Bits</th>
<th>Octet</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>transportSpecific</td>
<td>messageType</td>
<td>1</td>
</tr>
<tr>
<td>Reserved</td>
<td>versionPTP</td>
<td>1</td>
</tr>
<tr>
<td>messageLength</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>domainNumber</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Reserved</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>flagField</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>correctionField</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Reserved</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>sourcePortIdentity</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>sequenceId</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>controlField(1)</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>logMessageInterval</td>
<td>1</td>
<td>33</td>
</tr>
</tbody>
</table>

1. The controlField is used in version 1. In version 2, the messageType field is used for detecting different message types.

Some fields of the Ethernet payload can be used to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP packets:

- PTP packets over IPv4
- PTP frames over IPv6
- PTP packets over Ethernet

PTP packets over IPv4

Table 632 provides information about the fields that are matched to control the snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format defined in Table 631.

Table 632. Message format defined in IEEE 1588-2008

<table>
<thead>
<tr>
<th>Matched field</th>
<th>Octet position</th>
<th>Matched value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC packet type</td>
<td>12, 13</td>
<td>0x0800</td>
<td>IPv4 datagram</td>
</tr>
<tr>
<td>IP version and header length</td>
<td>14</td>
<td>0x45</td>
<td>IP version is IPv4</td>
</tr>
<tr>
<td>Layer 4 protocol</td>
<td>23</td>
<td>0x11</td>
<td>UDP</td>
</tr>
</tbody>
</table>
### Table 632. Message format defined in IEEE 1588-2008 (continued)

<table>
<thead>
<tr>
<th>Matched field</th>
<th>Octet position</th>
<th>Matched value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP multicast address (IEEE 1588 version 1)</td>
<td>30, 31, 32, 33</td>
<td>0xE0, 0x00, 0x01, 0x81 (or 0x82 or 0x83 or 0x84)</td>
<td>Multicast IPv4 addresses allowed: 224.0.1.129, 224.0.1.130, 224.0.1.131, 224.0.1.132</td>
</tr>
<tr>
<td>IP multicast address (IEEE 1588 version 2)</td>
<td>30, 31, 32, 33</td>
<td>0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)</td>
<td>PTP Primary multicast address: 224.0.1.129  PTP Pdelay multicast address: 224.0.0.107</td>
</tr>
<tr>
<td>UDP destination port</td>
<td>36, 37</td>
<td>0x013F, 0x0140</td>
<td>0x013F: PTP event messages (1) 0x0140: PTP general messages</td>
</tr>
<tr>
<td>PTP control field (IEEE 1588 version 1)</td>
<td>74</td>
<td>0x00, 0x01, 0x02, 0x03, 0x04</td>
<td>0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management</td>
</tr>
<tr>
<td>PTP message type field (IEEE 1588 version 2)</td>
<td>42 (nibble)</td>
<td>0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, 0xD</td>
<td>0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management</td>
</tr>
<tr>
<td>PTP version</td>
<td>43 (nibble)</td>
<td>0x1 or 0x2</td>
<td>0x1: Supports PTP version 1 0x2: Supports PTP version 2</td>
</tr>
</tbody>
</table>

1. PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only)

---

**PTP frames over IPv6**

*Table 633* provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format defined in *Table 631*.

### Table 633. IPv6-UDP PTP packet fields required for control and status

<table>
<thead>
<tr>
<th>Matched field</th>
<th>Octet position</th>
<th>Matched value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC packet type</td>
<td>12, 13</td>
<td>0x86DD</td>
<td>IP datagram</td>
</tr>
<tr>
<td>IP version</td>
<td>14 (bits [7:4])</td>
<td>0x6</td>
<td>IP version is IPv6</td>
</tr>
<tr>
<td>Layer 4 protocol</td>
<td>20(1)</td>
<td>0x11</td>
<td>UDP</td>
</tr>
</tbody>
</table>

---

(1) Layer 4 protocol: 0x21 for IPv6, 0x08 for IPv4.
PTP packets over Ethernet

Table 634 provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format.

<table>
<thead>
<tr>
<th>Matched field</th>
<th>Octet position</th>
<th>Matched value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTP multicast address</td>
<td>38 – 53</td>
<td>FF0:0:0:0:0:0:0:181</td>
<td>PTP primary multicast address: FF0x:0:0:0:0:0:0:181 (Hex)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FF02:0:0:0:0:0:0:6B</td>
<td>PTP Pdelay multicast address: FF02:0:0:0:0:0:0:6B (Hex)</td>
</tr>
<tr>
<td>UDP destination port</td>
<td>56, 57a</td>
<td>0x013F, 0x140</td>
<td>0x013F: PTP event message</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0140: PTP general messages</td>
</tr>
<tr>
<td>PTP control field (IEEE 1588 version 1)</td>
<td>94a</td>
<td>0x00, 0x01, 0x02, 0x03, or 0x04</td>
<td>0x00: SYNC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x01: Delay_Req</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x02: Delay_Resp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x03: Delay_Resp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x04: Management (version1)</td>
</tr>
<tr>
<td>PTP message type field</td>
<td>62a (nibble)</td>
<td>0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD</td>
<td>0x0: SYNC</td>
</tr>
<tr>
<td>(IEEE 1588 version 2)</td>
<td></td>
<td></td>
<td>0x1: Delay_Req</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x2: Pdelay_Req</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x3: Pdelay_Resp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x8: Follow_Up</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x9: Delay_Resp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0xA: Pdelay_Resp_Follow_Up</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0xB: Announce</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0xC: Signaling</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0xD: Management</td>
</tr>
<tr>
<td>PTP Version</td>
<td>63 (nibble)</td>
<td>0x1 or 0x2</td>
<td>0x1: Supports PTP version 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x2: Supports PTP version 2</td>
</tr>
</tbody>
</table>

1. The Extension header is not defined for PTP packets.
Transmit path functions

The MAC captures a timestamp when the start packet delimiter (SFD) of a packet is sent on the MII interface. The packets, for which a timestamp has to be captured, can be controlled on per-packet basis. Each Transmit packet can be marked to indicate whether a timestamp should be captured for it.

The MAC does not process the transmitted packets to identify the PTP packets. The packets for which a timestamp has to be captured must be specified. The packets can be defined by using the control bits in the Transmit Descriptor (see Section 57.10.3: Transmit descriptor). The MAC returns the timestamp to the software inside the corresponding Transmit descriptor, thus automatically connecting the timestamp to the specific PTP packet.

The 64-bit timestamp information is written to the TDES0 and TDES1 fields. The TDES0 field holds the 32 least significant bits of the timestamp.

Receive path functions

The MAC can be programmed to capture the timestamp of all packets received on the MII interface or to process packets to identify the valid PTP messages. The snapshot of the time to be sent to the application can be controlled by using the following options of the Timestamp control Register (ETH_MACTSCR):

- Enable snapshot for all packets
- Enable snapshot for IEEE 1588 version 1 or version 2 timestamp

<table>
<thead>
<tr>
<th>Matched field</th>
<th>Octet position</th>
<th>Matched value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTP control field (IEEE 1588 version 1)</td>
<td>46</td>
<td>0x00, 0x01, 0x02, 0x03, or 0x04</td>
<td>0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management</td>
</tr>
<tr>
<td>PTP message type field (IEEE 1588 version 2)</td>
<td>14 (nibble)</td>
<td>0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD</td>
<td>0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management</td>
</tr>
<tr>
<td>PTP version</td>
<td>15 (nibble)</td>
<td>0x1 or 0x2</td>
<td>0x1: Supports PTP version 1 0x2: Supports PTP version 2</td>
</tr>
</tbody>
</table>

1. The unicast address match of destination addresses (DA), programmed in MAC address 0 to 31, is used if the TSENMACADDR bit of Timestamp control Register (ETH_MACTSCR) is set.
2. IEEE 1588-2008, Annex F
3. The MAC does not consider the PTP version 1 messages with Peer delay multicast address (01-80-C2-00-00-0E) as valid PTP messages.
• Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet
• Enable timestamp snapshot for the received packet for IPv4 or IPv6
• Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP)
• Enable the node to be a master or slave and select the snapshot type
  This feature controls the type of messages for which snapshots are taken.

Note: The peripheral also supports the PTP messages over VLAN packets.

Table 635 indicates the PTP messages for which a snapshot is taken depending on the SNAPTYPESEL field in Timestamp control Register (ETH_MACTSCR).

<table>
<thead>
<tr>
<th>SNAPTYPESEL</th>
<th>TSMSTRENA</th>
<th>TSEVNTENA</th>
<th>PTP messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>X</td>
<td>0</td>
<td>SYNC, Follow_Up, Delay_Req, Delay_Resp</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>SYNC</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>Delay_Req</td>
</tr>
<tr>
<td>01</td>
<td>X</td>
<td>0</td>
<td>SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>1</td>
<td>SYNC, Pdelay_Req, Pdelay_Resp</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>1</td>
<td>Delay_Req, Pdelay_Req, Pdelay_Resp</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td>X</td>
<td>SYNC, Delay_Req</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
<td>X</td>
<td>Pdelay_Req, Pdelay_Resp</td>
</tr>
</tbody>
</table>

The DMA returns the timestamp to the software application inside the corresponding Receive descriptor. The extended status, containing the timestamp message status and the IPC status, is written in the RDES1 normal descriptor and the snapshot of the timestamp is written in RDES0 and RDES1 fields of the context descriptor. The RDES0 field holds the 32 least significant bits of the timestamp.

Programming guidelines for IEEE 1588 timestamping (system time correction)

See Section : System time correction in Section 57.9.9: Programming guidelines for IEEE 1588 timestamping on page 2769.

IEEE 1588 system time source

To get a snapshot of the time, the MAC requires a reference time in 64-bit format as defined in the IEEE 1588-2002 (80-bit format as defined in the IEEE 1588-2008).

Description of IEEE 1588 system time source

The peripheral uses the reference clock input and uses it to internally generate the Reference time (also called the system time) and capture timestamps.

The timestamp has the following fields:
• UInteger48 seconds field
The seconds field is the integer portion of the timestamp in units of seconds. It is 48-bit wide. For example, 2.000000001 seconds are represented as seconds Field = 0x0000 0000 0002.

- **UInteger32 nanosecondsField**
  The nanoseconds field is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 seconds are represented as nanoSeconds = 0x0000 0001.
  The nanoseconds field supports the following two modes:
  - **Digital rollover mode**: In this mode, the maximum value in the nanoseconds field is 0x3B9A C9FF, that is, (10e9-1) nanoseconds.
  - **Binary rollover mode**: In this mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF FFFF. Accuracy is ~0.466 ns per bit.
  These modes can be set through TSCTRLSSR bit in Timestamp control Register (ETH_MACTSCR).

### System time register module

The 64-bit PTP time is updated using the PTP input reference clock, clk_ptp_ref_i. This PTP time is used as a source to take snapshots (timestamps) of the Ethernet frames being transmitted or received at the MII.

The system time counter can be initialized or corrected using either the coarse or the fine correction method.

In the coarse correction method, the initial value or the offset value is written to the timestamp update register. For initialization, the system time counter is programmed with the value in the timestamp update registers, whereas for system time correction the offset value (timestamp update register) is added to or subtracted from the system time.

In the fine correction method, the slave clock (reference clock) frequency drift with respect to the master clock (as defined in IEEE 1588-2002 specifications) is corrected over a period of time, unlike in the coarse correction method where it is corrected in a single clock cycle. The longer correction time helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register as shown in Figure 802. The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. The accumulator acts as a high-precision frequency multiplier or divider.

This system time update algorithm is shown in Figure 802.
The system time update logic requires a 50 MHz clock frequency to achieve 20 ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. For example, if the reference clock (clk_ptp_ref_i) is 66 MHz, this ratio is calculated as 66 MHz/50 MHz = 1.32. Therefore, the default addend value to be set in the register is $2^{32} / 1.32$, or 0xC1F07C1F.

If the reference clock drifts lower, for example, to 65 MHz, the ratio is 65 / 50, that is 1.3 and the value to set in the addend register is $2^{32} / 1.30$, or 0xC4EC4EC4.

If the clock drifts higher, for example to 67 MHz, the addend register must be set to 0xBF0B7672. When there is not clock drift, the default addend value of 0xC1F07C1F ($2^{32} / 1.32$) must be programmed.

In Figure 802, the constant value used to accumulate the subsecond register is decimal 43, which achieves a system time accuracy of 20 ns (in other words, it is incremented in 20 ns steps).

The software must calculate the drift in frequency based on the SYNC messages and accordingly update the Addend register.

Initially, the slave clock is set with FreqCompensationValue0 in the Addend register. This value is as follows:

$$\text{FreqCompensationValue}_0 = \frac{2^{32}}{\text{FreqDivisionRatio}}$$
If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm given in this section must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and resynchronize with the master using the new value.

The algorithm is as follows:

1. At time MasterSyncTime\(_n\) the master sends the slave clock a SYNC message. The slave receives this message when its local clock is SlaveClockTime\(_n\) and computes MasterClockTime\(_n\) as follows:
   \[
   \text{MasterClockTime}_{n} = \text{MasterSyncTime}_{n} + \text{MasterToSlaveDelay}_{n}
   \]

2. The master clock counts for current Sync cycle, MasterClockCount\(_n\) is
   \[
   \text{MasterClockCount}_{n} = \text{MasterClockTime}_{n} - \text{MasterClockTime}_{n-1}
   \]
   (assuming that MasterToSlaveDelay is the same for Sync cycles \(n\) and \(n - 1\))

3. The slave clock count for current Sync cycle, SlaveClockCount\(_n\) is
   \[
   \text{SlaveClockCount}_{n} = \text{SlaveClockTime}_{n} - \text{SlaveClockTime}_{n-1}
   \]

4. The difference between master and slave clock counts for current Sync cycle, ClockDiffCount\(_n\) is
   \[
   \text{ClockDiffCount}_{n} = \text{MasterClockTime}_{n} - \text{SlaveClockTime}_{n}
   \]

5. The frequency-scaling factor for slave clock, FreqScaleFactor\(_n\) is
   \[
   \text{FreqScaleFactor}_{n} = (\text{MasterClockCount}_{n} + \text{ClockDiffCount}_{n})/ \text{SlaveClockCount}_{n}
   \]

6. The frequency compensation value for Addend register, FreqCompensationValue\(_n\) is
   \[
   \text{FreqCompensationValue}_{n} = \text{FreqScaleFactor}_{n} \times \text{FreqCompensationValue}_{n-1}
   \]

In theory, this algorithm achieves the lock in one Sync cycle. However, it may take several cycles, because of changing network propagation delays and operating conditions. This algorithm is self-correcting. If the slave clock is initially set to an incorrect value by the master, the algorithm corrects it at the cost of additional Sync cycles.

Refer to Section 57.9.9: Programming guidelines for IEEE 1588 timestamping for detailed programming steps.

**IEEE 1588 auxiliary snapshots**

The auxiliary snapshot feature enables to store a snapshot of the system time based on an external event. The event is considered to be the rising edge of the eth_ptp_trgx (where \(x = 1\) to 4) sideband signal.

Up to four auxiliary snapshot inputs can be configured and up to four snapshots can be stored. A FIFO is accessible through registers: Auxiliary timestamp seconds register (ETH_MACATSSR) and Auxiliary timestamp nanoseconds register (ETH_MACATSNR).

The snapshots taken for any input are stored in a common FIFO; only 64 bits are kept. The application can read the Timestamp status register (ETH_MACTSSR) to know the timestamp of which input is available for reading at the top of this FIFO.
When a snapshot is stored, the MAC indicates this to the application with an interrupt. The value of the snapshot is read through a FIFO register access. If the FIFO becomes full and an external trigger to take the snapshot is asserted, a snapshot trigger-missed status (ATSSTM) is set in the Timestamp status register (ETH_MACTSSR). This indicates that the latest auxiliary snapshot of the timestamp is not stored in the FIFO. The latest snapshot is not written to the FIFO when it is full.

When an application reads the 64-bit timestamp from the FIFO, the space becomes available to store the next snapshot. You can clear a FIFO by setting the ATSFC bit in Auxiliary control register (ETH_MACACR). When multiple snapshots are present in the FIFO, the count is indicated in bits[27:25] of Timestamp status register (ETH_MACTSSR).

**Flexible pulse-per-second output**

The MAC supports either a fixed pulse-per-second output mode (also called fixed mode) or a flexible pulse-per-second output mode for the ETH_PPS_OUT and eth_ptp_pps_out outputs:

- **Fixed pulse-per-second output**
  
  In this mode, only the frequency of the PPS output can be changed by setting the PPSCTRL0 field in the PPS control register (ETH_MACPPSCR).

- **Flexible pulse-per-second output**
  
  In this mode, the software has the flexibility to program the start or stop time, width, and interval of the pulse generated on the eth_ptp_pps_out output:
  
  The start and stop times are programmed through PPS target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTTNR).
  
  The PPS width and interval are programmed in terms of granularity of system time (number of the units of subsecond increment value) through PPS width register (ETH_MACPPSWR) and PPS interval register (ETH_MACPPSIR), respectively.

*Note:* By default, the peripheral is in Fixed mode and indicates one second interval. When Fixed mode is selected by clearing PPSEN0 to 0 in the PPS control register (ETH_MACPPSCR):

- the output on all PPS outputs is controlled by the value programmed in the PPSCTRL_PPSCMD field. Independent control of individual PPS output is not supported in Fixed mode.

- PPS target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTTNR) are used only for generating target time reached interrupt; they are not used for PPS output generation.

- TRGTMODSEL0/1/2/3 must be programmed to 0.

- the frequency of the PPS output can be changed by setting the PPSCTRL0 field in the PPS control register (ETH_MACPPSCR).
Description of flexible pulse-per-second (PPS) output

The peripheral supports the following features with the flexible PPS outputs:

- Programming the start or stop time in terms of system time.
- Programming the start point of the single pulse and start and stop points of the pulse train in terms of 64-bit system time. The Target Time registers are used to program the start and stop time.
- Programming the stop time in advance, that is, the stop time can be programmed before the actual start time has elapsed.
- Programming the width between the rising edge and corresponding falling edge of PPS signal output in terms of number of units of subsecond increment value programmed in the Subsecond increment register (ETH_MACSSIR). The pulse width can be programmed from 1 to 2^{32}–1 units of subsecond increment value.
- Programming the interval, between the rising edges of PPS signal, in terms of number of units of subsecond increment value. You can program the interval between pulses from 1 to 2^{32}–1 units of subsecond increment value.
- Option to cancel the programmed PPS start or stop request.
- Error if the start or stop time being programmed has already elapsed.

Note: The PTP reference clock mentioned in the following sections is the clock at which the system time is updated. When the TSCFUPDT bit of Timestamp control Register (ETH_MACTSCR) is set to 0, this clock is similar to the clk_ptp_ref_i clock. In Fine correction mode, this is the clock tick at which the system time is updated (using Subsecond increment register (ETH_MACSSIR)) (as shown in Figure 802).

Refer to Section 57.9.12: Programming guidelines for flexible pulse-per-second (PPS) output for further details on how configuring flexible pulse output.

PPS start and stop times

The initial start time can be programmed in the Target Time registers.

If required, the start or stop time can be programmed again. However, this can be done only after the earlier programmed value is synchronized with the PTP clock domain. Bit 31 of PPS target time nanoseconds register (ETH_MACPPSTTNR) indicates that the synchronization is complete. This enables to program the start or stop time in advance even before the earlier stop or start time has elapsed.

To ensure proper PPS signal output, it is recommended to program advanced system time for the start or stop time. If the application programs a start or stop time that has already elapsed, the MAC sets an error status bit indicating the programming error. If enabled, the MAC also sets the Target Time Reached interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

PPS width and interval

The PPS width and interval are programmed in terms of granularity of system time, that is, number of the units of subsecond increment value. For example, to obtain a PPS pulse width of 40 ns and an interval of 100 ns with a PTP reference clock of 50 MHz, program the width and interval to values 2 and 5, respectively. Smaller granularity can be achieved by using a faster PTP reference clock.

Before giving the command to trigger a pulse or pulse train on the PPS output, program or update the interval and width of the PPS signal output.
PTP timestamp offload function

This feature enables the automatic generation of specific PTP packets to be performed, when the MAC operates as a specific node in the PTP network.

These packets can be generated periodically or triggered by the host software. In other modes, this feature can parse the incoming PTP packets on the receiver, and automatically generate and respond to the required PTP packets. It helps to offload certain PTP node functions with better accuracy and lower response latency.

The PTP offload feature is selected through **PTP Offload control register (ETH_MACPOCR)**. 80-bit PTP node identity is configured through the following three registers: **PTP Source Port Identity 0 Register (ETH_MACSPI0R)**, **PTP Source port identity 1 register (ETH_MACSPI1R)** and **PTP Source port identity 2 register (ETH_MACSPI2R)**.

Description of PTP offload function

Depending on the programmed mode, the MAC generates PTP Ethernet messages periodically or from the application, or based on reception of a particular PTP message. **Table 636** indicates the PTP message generation criteria.

<table>
<thead>
<tr>
<th>SNAPYPSEL</th>
<th>TSMSTRENA</th>
<th>TSEVNTENA</th>
<th>Mode</th>
<th>Criteria for generation of PTP messages</th>
<th>PTP message type generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>Ordinary or Boundary Slave</td>
<td>SYNC message reception</td>
<td>Delay_Req</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>Ordinary or Boundary Master</td>
<td>Periodic or on trigger from application</td>
<td>SYNC</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>Transparent Slave</td>
<td>Pdelay_Req</td>
<td>Pdelay_Resp</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>1</td>
<td>Transparent Slave</td>
<td>SYNC message reception</td>
<td>Delay_Req</td>
</tr>
</tbody>
</table>
Note: Clocks supporting peer delay mechanism must not generate delay request/delay response messages, according to IEEE 1588-2008 specifications. However, the peripheral controller supports this for flexibility, with a programmable control bit (DRRDIS) in the PTP Offload control register (ETH_MACPOCR).

The DRRDIS bit can be used to control the response generation for delay request/delay response message. For example, in transparent slave mode, delay request is generated in response to received \texttt{SYNC} only when the bit is reset.

When the MAC is set as an Ordinary or Boundary Slave clock in the PTP network, it can respond to the reception of \texttt{SYNC} messages with an automatic generation and transmission of the corresponding Delay\_Req message. Similarly, various other modes of operation are explained in Table 636.

The MAC supports the multicast communication model for the generation of \texttt{SYNC} and Pdelay\_Req PTP messages. For instance, the Destination Address field of the generated PTP over Ethernet packet is the defined special multicast addresses (0x011B 1900 0000 for all except peer delay mechanism messages and 0x0180 C200 000E for peer delay mechanism messages).

When the MAC responds to received \texttt{SYNC}, Delay\_Req and Pdelay\_Req PTP messages with special multicast destination address, it also uses the corresponding special multicast address in the DA field of the automatically generated Delay\_Req, Delay\_Resp, and Pdelay\_Resp PTP messages, respectively.

When the MAC responds to received \texttt{SYNC}, Delay\_Req and Pdelay\_Req PTP messages with unicast destination address, it takes the SA field of the received packets and makes

<table>
<thead>
<tr>
<th>Programming</th>
<th>Mode</th>
<th>Criteria for generation of PTP messages</th>
<th>PTP message type generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Transparent Master</td>
<td>Periodic or on trigger from application</td>
<td>Pdelay_Req</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pdelay_Req message reception</td>
<td>Pdelay_Resp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Periodic or on trigger from application</td>
<td>SYNC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay_Req message reception</td>
<td>Delay_Resp</td>
</tr>
<tr>
<td>11</td>
<td>Peer-to-Peer Transparent</td>
<td>Periodic or on trigger from application</td>
<td>Pdelay_Req</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pdelay_Req message reception</td>
<td>Pdelay_Resp</td>
</tr>
</tbody>
</table>

All other programming combinations are invalid for PTP Offload feature.
them as the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

At the same time, all the received PTP messages are forwarded to the application along with Rx status, indicating whether the response was generated by the MAC, if it satisfies the packet filtering logic of the MAC receiver.

When the MAC automatically generates a PdelayReq or responds with a Delay_Req, the egress timestamp of these two PTP messages are provided in the Tx TS status (Tx Timestamp Status register and interrupt generated).

In addition to messageType and versionPTP fields match for basic PTP over Ethernet message detection, the following additional fields are matched to qualify the received PTP message type:

1. The domainNumber field is checked for a match against the value programmed in the CSR.
2. The twoStepFlag in flagField field is checked for one-step indication (0b0).
3. The transportSpecific field is checked for Default PTP over Ethernet (0b0000) or 802.1AS mode (0b1111) when enabled.

**PTP packet generation**

This section explains the format and content of the automatically generated PTP packets by the MAC when this mode is enabled. It provides the template of the common PTP message header, as well as the detailed description of the fields of the specific PTP packets generated.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Octets</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>transportSpecific messageType</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Reserved versionPTP</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>messageLength</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>domainNumber</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Reserved</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>flagField</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>correctionField</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Reserved</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>sourcePortIdentity</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>sequenceId</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>sequenceId</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>controlField</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>logMessageInterval</td>
<td>1</td>
<td>33</td>
</tr>
</tbody>
</table>

---

Table 637. Common PTP message header fields
PTP message header fields

- **messageType**
  The following encoded values are used for PTP message types:
  - SYNC: 0000
  - Delay_Req: 0001
  - Pdelay_Req: 0010
  - Pdelay_Resp: 0011
  - Delay_Resp: 1001

- **transportSpecific**
  The following transport protocol encoding is used:
  - Default PTP over Ethernet: 0000
  - 802.1AS mode: 0001

- **versionPTP**
  It is always set to 2 because PTP version 2 is supported.

- **domainNumber**
  This field contains the value from the **PTP Offload control register (ETH_MACPOCR)**.

- **flagField**
  The following values are used:
  - alternateMasterFlag (Octet 0 bit 0): 0 for SYNC and Delay_Resp
  - twoStepFlag (Octet 0 bit 1): 0 for SYNC and Pdelay_Resp
  - unicastFlag (Octet 0 bit 2): 0 for Multicast Address, 1 for Unicast Address

- **correctionField**
  For more information, see **Table 638**.

- **sourcePortIdentity**
  This field takes the value programmed in the **PTP Source Port Identity 0 Register (ETH_MACSPI0R)**, **PTP Source port identity 1 register (ETH_MACSPI1R)** and **PTP Source port identity 2 register (ETH_MACSPI2R)**.

- **sequenceId**
  Pdelay_Resp and Delay_Resp use the same sequenceId field from received Pdelay_Req and Delay_Req PTP messages. For SYNC/Delay_Req, Pdelay_Req, a separate sequenceId counter is maintained. These sequenceId counters get incremented by 1 every time the corresponding message is generated and transmitted.

- **controlField**
  The following encoded values are used for controlField:
  - SYNC: 0000 0000
  - Delay_Req: 0000 0001
  - Pdelay_Req: 0000 0010
  - Pdelay_Resp: 0000 0101
  - Delay_Resp: 0000 0011

- **logMessageInterval**
  - SYNC:
    This field contains logSyncInterval from the corresponding MAC_Log_Message_Interval register.
DelayResp:  
This field contains the sum of DRSYNC and logSyncInterval value taken from  
the Log message interval register (ETH_MACLMIR) for a multicast PTP message  
and 0111 1111 for unicast PTP message.

DelayReq, PdelayReq and PdelayResp: 0111 1111  
where logSyncInterval = log2 (Mean Value of Interval in seconds)

The MAC supports values of –15 to 15 for logSyncInterval fields, which translates to a range  
from 32.768 micro second (2–15) to 215 second. For a given value of log sync interval (N),  
the time interval between two SYNC packets is given by the following:
- \[2^{(30+N)} \text{ ns}, \text{ when } N \text{ is negative } (-1 \text{ to } -15)\]
- \[2^N \text{ seconds}, \text{ when } N \text{ is positive } (0 \text{ to } 15)\]

For example:
- When logSyncInterval is programmed to 1, the interval is \(2^1\); therefore, the SYNC  
message is sent once every 2 seconds.
- When logSyncInterval is programmed to -1, the interval is \(2^{-1} = 0.536 \text{ seconds}\);  
therefore, the SYNC message is sent once every 536 milliseconds. The value is 0.536  
seconds, because \(2^{-30} = 1 \text{ ns}\).
- When logSyncInterval is programmed to –5, the interval is \(2^{-5} = 33.55 \text{ ms}\); therefore,  
the SYNC message is sent once every 33.55 ms.

Note: The MAC uses the PTP system time to generate the intervals for periodic packet  
transmission. For negative values of log message interval programmed, the generated  
period may deviate from the value given by the equation \(2^{(30+N)}\), because of the non-binary  
nature of the nanoseconds field of the system time.

PTP message-specific fields

The message-specific fields are the following:
- **messageLength**  
There is no suffix supported, so this field contains the length of the PTP message that  
includes 34-byte PTP common header and the body specific to the message type.  
For SYNC and Delay_Req packets, this field contains 44, whereas for Delay_Resp,  
Pdelay_Req and Pdelay_Resp, it contains 54.
- **originTimestamp**  
This field is the captured egress timestamp for SYNC, Delay_Req, and Pdelay_Req  
PTP messages.
- **receiveTimestamp**  
For Delay_Resp PTP message, this is the ingress timestamp of the corresponding  
received Delay_Req PTP message.
- **requestingPortIdentity**  
For Delay_Resp and Pdelay_Resp PTP messages, this is the sourcePortIdentity field  
taken from the corresponding received Delay_Req and Pdelay_Req PTP messages.
- **requestReceiptTimestamp**  
For the Pdelay_Resp PTP message, this field is set to 0.
One-step timestamp

The MAC supports the one-step timestamp feature that enables to identify the offset in the packet and inserts the timestamp received from the application at that offset.

MAC Transmit PTP mode for one-step timestamp

Depending upon the type of message and its mode, the MAC updates the following fields of Transmit PTP packets:

- correctionField in the PTP header of messages
- originTimestamp in SYNC, Delay_Req, and Pdelay_Req messages

Table 638 shows how the PTP mode is selected based on the settings of SNAPTYPSel, TSMSTRENA, and TSEVNTENA bits of the Timestamp control Register (ETH_MACTSCR) and the fields that are updated for the incoming PTP packets based on the message type in that mode, during the one-step timestamping operation.

Table 638. MAC Transmit PTP mode and one-step timestamping operation

<table>
<thead>
<tr>
<th>Programming</th>
<th>Mode</th>
<th>Per packet control(1)</th>
<th>Messages processed on transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNAPTYPSel</td>
<td>TSMSTR</td>
<td>TSEVNT</td>
<td>TTSE</td>
</tr>
<tr>
<td>ENA</td>
<td>ENA</td>
<td>ENA</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td>00</td>
<td>X</td>
<td>0</td>
<td>End-to-end transparent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>Ordinary or Boundary Slave</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>Ordinary or Boundary Master</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>X</td>
<td>0</td>
<td>End-to-end Transparent with support for peer delay mechanism</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 638. MAC Transmit PTP mode and one-step timestamping operation (continued)

<table>
<thead>
<tr>
<th>Programming</th>
<th>Mode</th>
<th>Per packet control(1)</th>
<th>Messages processed on transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNAPTYPE SEL</td>
<td>TSMSTR ENA</td>
<td>TSEVNT ENA</td>
<td>TTSE (2)</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ordinary or Boundary Slave with support for peer delay mechanism or Peer-to-peer transparent</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ordinary or Boundary Master with support for peer delay mechanism</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td>X</td>
<td>End-to-end transparent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 638. MAC Transmit PTP mode and one-step timestamping operation (continued)

<table>
<thead>
<tr>
<th>Programming</th>
<th>Mode</th>
<th>Per packet control(1)</th>
<th>Messages processed on transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TTSE (2)</td>
<td>OSTC (3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

1. The per-packet control values provided here are the recommended settings used by devices in typical PTP operation and for the programmed mode.
2. TTSE represents TTSE bit of TDES2 transmit normal descriptor. The TTSE function is independent from the OST function and the programmed operation mode for OST. The MAC captures and returns the timestamp when the TTSE bit is set.
3. OSTC represents OSTC bit of TDES3 transmit context descriptor
4. TTS represents the timestamp value provided in the TTSH, TTSL fields of TDES0 and TDES1 transmit normal descriptor (write-back format).

**Note:** Residence time/turnaround time is calculated as the difference between the captured timestamp (egress timestamp) and the ingress timestamp. Clocks supporting peer delay mechanism do not use delay request or response, but it is included in OST for flexibility.

**Enabling one-step timestamp**

The one-step timestamp feature can be enabled for a given packet by setting bit 20 (OSTC) in TDES3 context descriptor. To update the correction field in certain PTP packets, the ingress timestamp must be given in the TSSL and TSSH fields.

The one-step timestamp feature is supported only for the PTP over Ethernet packets. It is not supported for PTP over IPv4/IPv6 packets.

**57.5.5 Checksum offload engine**

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. The MAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the Transmit path, as well as error detection in the Receive path.

**Transmit checksum offload engine**

In the transmit path, the MAC calculates the checksum and inserts it in the Tx packet. This feature helps reducing the load on the software and can improve the overall system throughput.

The COE module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 bits[17:16]).
Note: The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field. Because of this requirement, the Tx FIFO automatically operates in the Store-and-forward mode even if the MAC is configured in Threshold (cut-through) mode.

Make sure that the Tx FIFO is deep enough to store a complete packet before that packet is transferred to the MAC transmitter, the reason being that when space is not available to accept the programmed burst length of data, then the MTL Tx FIFO starts reading to avoid deadlock. In such a case, the COE fails as the start of the packet header is read out before the payload checksum can be calculated and inserted. Therefore, the checksum insertion must be enabled only in the packets that are less than the number of bytes, given by the following equation:

\[
\text{Packet size} < \text{TxQSize} - (\text{PBL} + 7) \times 4
\]

where

- \(\text{TxQSize}\) corresponds to the TQS bitfield of `Tx queue operating mode Register (ETH_MTLTXQOMR)`
- \(\text{PBL}\) corresponds to the TxPBL bitfield of `Channel transmit control register (ETH_DMACTXCR)`

Refer to IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460, and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6, and ICMPv6 packet header specifications, respectively.

**IP header checksum engine**

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Type field of Ethernet packet has the value 0x0800 and the Version field of IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

**Note:** IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status in the Transmit status (bit 0 in Table 660: TDES3 normal descriptor (write-back format)).

This status bit is set whenever the values of the Ethernet Type field and the Version field of IP header are not consistent, or when the Ethernet packet does not have enough data, as
indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The received Ethernet type is 0x0800, but the Version field of IP header is not equal to 0x4.
  - The IPv4 Header Length field indicates a value less than 0x5 (20 bytes).
  - The total packet length is less than the value given in the IPv4 Header Length field.
- For IPv6 datagrams:
  - The Ethernet type is 0x86DD but the IP header Version field is not equal to 0x6.
  - The packet ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

**TCP/UDP/ICMP checksum engine**

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The Tx COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input packet. This engine includes the Checksum field in the checksum calculation, and then replaces the Checksum field with the final calculated checksum.
- The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

**Note:** For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 0x0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 0x0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (bit 12 in Table 660: TDES3 normal descriptor (write-back format)). This engine sets the Payload Checksum Error status bit when it detects that the packet has been forwarded to the MAC Transmitter engine in the store-and-forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

*Table 639* describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as “No” in the table.

**Note:** Do not enable checksum insertion for IPv4 or IPv6 packets that are greater than the frame size constraint specified in *Section: Transmit checksum offload engine* because it might result in incorrect checksum insertion or unexpected behavior.
<table>
<thead>
<tr>
<th>Packet type</th>
<th>Hardware IP header checksum insertion</th>
<th>Hardware TCP/UDP checksum insertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-IPv4 or IPv6 packet</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in the MAC) but less than or equal to the frame size constraint specified in <a href="#">Section : Transmit checksum offload engine.</a></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in MAC) but less than or equal to the frame size constraint specified in <a href="#">Section : Transmit checksum offload engine.</a></td>
<td>Not applicable</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 with TCP, UDP, or ICMP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 packet has IP options (IP header is longer than 20 bytes)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Packet is an IPv4 fragment</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IPv4 packet with the following next header fields in main or extension headers:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>– Hop-by-hop options (in IPv6 main header)</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– Hop-by-hop options (in IPv6 extension header)</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>– Destinations options</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– Routing (with segment left 0)</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>– Routing (with segment left &gt; 0)</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– TCP, UDP, or ICMP</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>– Authentication</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– Any other next header field in main or extension headers</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>IPv4 packet has TCP header with Options fields</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 Tunnels:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>– IPv4 packet in an IPv4 tunnel</td>
<td>– Yes (IPv4 tunnel header)</td>
<td>– No</td>
</tr>
<tr>
<td>– IPv6 packet in an IPv4 tunnel</td>
<td>– Yes (IPv4 tunnel header)</td>
<td>– No</td>
</tr>
<tr>
<td>IPv6 Tunnels:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>– IPv4 packet in an IPv6 tunnel</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>– IPv6 packet in an IPv6 tunnel</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>IPv4 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled).</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv6 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled).</td>
<td>Not applicable</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 frames with security features (such as encapsulated security payload)</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IPv6 frames with security features (such as encapsulated security payload)</td>
<td>Not applicable</td>
<td>No</td>
</tr>
</tbody>
</table>
Receive checksum offload engine

The Receive Checksum Offload engine is used to detect errors in IP packets by calculating the header checksum and further matching it with the received header checksum. This engine also identifies a TCP, UDP, or ICMP payload in received IP packets and calculates the checksum of such payloads appropriately.

The Receive Checksum Offload Engine (Rx COE) can be enabled by setting the IPC bit of Operating mode configuration register (ETH_MACCR). When this bit is set, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the Type field of the received Ethernet packet. This identification is applicable to single VLAN-tagged packets. It is also applicable to double VLAN-tagged packets when the EDVLP bit of the VLAN tag register (ETH_MACVTR) is set.

The Rx COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

Packets with TCP/IP errors (header or payload) are dropped in MTL when DIS_TCP_EF bit of the Rx queue operating mode register (ETH_MTLRXQOMR) is reset and FEP bit is set.

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

Table 640: Receive checksum offload engine functions for different packet types describes the functions supported by the Rx COE based on the packet type. When the payload of an IP packet is not processed (indicated as "No" in the table), the information (whether the checksum engine is bypassed or not) is given in the receive status.

Note: The MAC does not append any payload checksum bytes to the received Ethernet packets.
### Table 640. Receive checksum offload engine functions for different packet types

<table>
<thead>
<tr>
<th>Packet type</th>
<th>Hardware IP header checksum checking</th>
<th>Hardware TCP/UDP/ICMP checksum checking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-IPv4 or IPv6</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in the MAC)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K packets is enabled in the MAC)</td>
<td>Not applicable</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 with TCP, UDP, or ICMP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IPv4 packet has IP options (IP header is longer than 20 bytes)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Packet is an IPv4 fragment</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IPv6 packet with the following next header fields in main or extension headers:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>– Hop-by-hop options (in IPv6 main header)</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– Hop-by-hop options (in IPv6 extension header)</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>– Destinations options</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– Routing (with segment left 0)</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– Routing (with segment left &gt; 0)</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– TCP, UDP, or ICMP</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– Any other next header field in main or extension headers</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>IPv4 packet has TCP header with Options fields</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 Tunnels:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>– IPv4 packet in an IPv4 tunnel</td>
<td>– Yes (IPv4 tunnel header)</td>
<td>– No</td>
</tr>
<tr>
<td>– IPv6 packet in an IPv4 tunnel</td>
<td>– Yes (IPv4 tunnel header)</td>
<td>– No</td>
</tr>
<tr>
<td>IPv6 Tunnels:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>– IPv4 packet in an IPv6 tunnel</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>– IPv6 packet in an IPv6 tunnel</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).</td>
<td>Not applicable</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 frames with security features (such as encapsulated security payload)</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IPv6 frames with security features (such as encapsulated security payload)</td>
<td>Not applicable</td>
<td>No</td>
</tr>
</tbody>
</table>
57.5.6 TCP segmentation offload

The MAC supports the TCP segmentation offload (TSO) feature in which the DMA splits a large TCP packet into multiple small packets and passes these packets to the MTL as shown in Figure 803.

This feature is enabled by programming the TSE bit of corresponding ETH_DMACCR register (see Channel transmit control register (ETH_DMACTXCR)). It is only supported when the MAC operates in Full-duplex mode.

For detailed programming steps, refer to Section 57.9.13: Programming guidelines for TSO.

Figure 803. TCP segmentation offload overview

![TCP segmentation offload overview diagram]
DMA operation with TSO feature

*Figure 804* shows the TSO flow.

*Figure 804. TCP segmentation offload flow*

![TCP segmentation offload flow diagram](image)

For the TSO feature, the Tx DMA operation is as follows:

1. The application sets up the Transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES3[31]) after setting up the corresponding data buffer(s) with Ethernet packet data.

2. The application increases the offset value of the Descriptor tail pointer of the DMA Tx channel.

3. While in the Run state, the DMA fetches the next available descriptor and performs one of the following actions:
   - If the descriptor is a context descriptor and the context is not between the first and last descriptors of a packet, the DMA stores the context values.
– If the descriptor is a context descriptor and the context is between the first and last descriptors of a packet, the DMA closes the context descriptor indicating a Context Descriptor Error (TDES3[23]) and fetches the next descriptor.

– If the descriptor is a normal descriptor, the DMA checks the TSE bit. If the TSE bit is not set, the DMA continues with the default mode of operation or OSF operation (if enabled).

4. The DMA calculates the number of segments from the TCP payload length (TDES3[17:0]) and the MSS value.

5. The DMA goes through channel arbitration to fetch the data buffers. The DMA fetches the header and payload separately.

6. For the first segment, the DMA fetches the header from the system memory and stores it in the TSO memory (if present and when the length of header is not greater than the TSO memory size). If the current segmented packet is not the first segment, it fetches again the header buffer in system memory, as done for the first segment. In such cases, the DMA does not close the first descriptor containing the header buffer until the header for last segment is fetched.

7. The required fields in the header bytes are modified/updated as per the segmentation requirements and written into the corresponding MTL Tx queue.

8. The DMA then takes the payload buffer pointer, fetches the MSS number of payload bytes from the system memory, and directly pushes it into the MTL Tx queue. If the buffer(s) in the descriptor do(es) not have enough data for the MSS payload (except for the last segment), the DMA closes this descriptor.

9. The DMA jumps to Step 3 and repeats the process until the last segment is written into the Tx queue.

10. The DMA closes the last descriptor and the first descriptor (containing the header buffer when it is not stored in TSO memory), and then moves on to the next packet transfer.

The DMA repeats all these steps if more descriptors are available. When no more descriptor are available, the DMA enters the suspend state.

Note: The TSO engine determines whether to perform TSO or USO operation based on the THL field (TCP Header Length) in TDES3 of first Normal Tx descriptor for the packet. The value of 2 indicates USO and any value greater than or equal to 5 indicates TSO.
TCP/IP header fields

While segmenting a TCP packet, the DMA automatically updates the TCP/IP header fields. *Table 641* describes how the TCP and IP headers are updated.

**Table 641. TSO: TCP and IP header fields**

<table>
<thead>
<tr>
<th>Packet sequence</th>
<th>TCP header</th>
<th>IP header</th>
</tr>
</thead>
</table>
| **First packet** | 1. The sequence number is not updated. The value provided in the header is used.  
2. If set, the FIN and PSH flags are cleared.  
3. The TCP checksum is calculated again. | IPv4 Header  
– Total Length = MSS + TCP Header Length + IP Header Length  
– Identification field is not modified. It is sent as per the header provided by the software.  
– IPv4 Header Checksum is recalculated.  
IPv6 Header  
– Payload Length = MSS + TCP Header Length + IP Extension Header Length |
| **Subsequent packets** | 1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment.  
2. If set, the FIN and PSH flags are cleared.  
3. The TCP checksum is calculated again. | IPv4 Header  
– Total Length = MSS + TCP Header Length + IP Header Length  
– Identification field = Previous Identification Field + 1  
– IPv4 Header Checksum is recalculated  
IPv6 Header  
– Payload Length = MSS + TCP Header Length + IP Extension Header Length |
| **Last packet** | 1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment.  
2. If FIN and PSH flags were set in original header, these flags are set.  
3. The TCP checksum is calculated again. | IPv4 Header  
– Total Length = Remaining Payload + TCP Header Length + IP Header Length  
– Identification Field = Previous Identification Field + 1  
– IPv4 header Checksum is recalculated  
IPv6 Header  
– Payload Length = Remaining Payload + TCP Header Length + IP Extension Header Length |
**Header and payload fields of segmented packets**

After segmentation, the split packets use the same header as the parent TCP packet for header fields other than the ones described in Table 641: TSO: TCP and IP header fields. Figure 805: Header and payload fields of segmented packets shows how same header is used for the header fields of segmented packets.

The application must create the header in Buffer 1 of the first descriptor of the packet to be segmented and provide the header length in TDES2 of the first descriptor (FD = 1). When the FD bit is set, the DMA reads the header from the header buffer to which the TDES0 is pointing. Buffer 2 of the first descriptor can be used for payload and TDES0 and TDES1 of subsequent descriptors. For subsequent descriptors (FD = 0), the address to which the TDES0 and TDES1 are pointing is treated as payload buffer address of the same packet.

**Context descriptor sequence**

The context descriptor can provide the maximum segment size (MSS) value for segmentation. The application must provide the context descriptor before the normal descriptor to be used for the corresponding TCP packet. If the application needs to provide a new MSS, it must create the context descriptor in the descriptor list before the first normal descriptor of the packet to be segmented with the new MSS value. The MSS value in the context descriptor is valid only if the TCMSSV bit of TDES3 in context descriptor is set and the OSTC bit is reset (refer to Section 57.10.3: Transmit descriptor).

When the application provides a context descriptor with a valid MSS value, the DMA internally stores the MSS value and uses this value for all subsequent packets for which the TSO is enabled through the TSE bit of TDES3 normal descriptor.
If the application places a context descriptor in the middle of a packet (between the first and last descriptors of a packet), the DMA does the following:

1. The DMA ignores the context and closes the descriptor.
2. The DMA indicates the error in descriptor status.
3. The DMA generates an interrupt if the CDEE bit is set in the Interrupt enable register corresponding to a DMA channel (see Channel interrupt enable register (ETH_DMACIER)).

The application can read the interrupt status through CDE bit of Status register corresponding to a DMA channel (see Channel status register (ETH_DMACSR)).

**Building the Descriptor and the packet for the TSO feature**

To enable segmentation for a packet, the application must set the TSE bit of TDES3 of first normal descriptor (see Section 57.10.3: Transmit descriptor). If the TSE bit is set in TDES3 for a non TCP/IP packet, the DMA behavior is unpredictable.

The application must program the length of the TCP packet payload in TDES3[17:0] and the TCP header in TDES3[22:19]. The maximum length of TCP packet payload that can be segmented is 256 Kbytes.

The header of the packet including Ethernet header, L3 header and L4 header should be provided in Buffer1 of the first normal descriptor of the TSO packet. Only buffer 1 of the first normal descriptor of a packet enabled for TSO is taken as the buffer containing the header.

The TCP payload can begin from buffer 2 of the first normal descriptor and continue to buffer 1 and buffer 2 of second normal descriptor and subsequent descriptors.

The TCP payload may span across multiple buffers and multiple descriptors. The size of buffers containing the TCP payload should add up to be equal to the TCP payload length provided in TDES3[17:0] of the first normal descriptor.

The MAC always calculates and appends CRC and inserts Padding (if required) for all packets segmented by the DMA. If the TSE bit of TDES3 is enabled, the CRC PAD Control (CPC) field of TDES3 is reserved. To determine the size of a TCP packet after segmentation, the DMA uses the Maximum Segment Size (MSS) provided by the application through context descriptor. The DMA segments only those packets which have payload size greater than MSS. The application must provide the MSS by either programming the MSS value in ETH_DMACCR (see Channel control register (ETH_DMACCR)) or by providing a context descriptor. The DMA uses the last programmed value of MSS or the last MSS value provided through context (whichever is provided later).

The header length plus the MSS size (which is equal to the size of each TCP segment) should not exceed 16383 bytes otherwise the MAC transmitter truncates the packet after 16383 bytes causing a CRC error.

The header length plus MSS size plus programmed PBL value in ETH_DMACTXCR register (see Channel transmit control register (ETH_DMACTXCR)) should be lesser than the Tx queue size programmed in TQS field of ETH_MTLTXQOMR register (see Tx queue operating mode Register (ETH_MTLTXQOMR)). A MSS plus header equal to half the programmed Tx queue size is recommended.

The DMA also supports segmentation of VLAN-tagged TCP/IP frames. If the TCP packet has a VLAN tag, then the same tag is used for all the segments irrespective of the VLAN tag type provided (C-VLAN or S-VLAN). The VLAN tag insert/replace control bits are used for all segments.
If the Double VLAN feature is selected, then the DMA passes both tags for all segments irrespective of the VLAN tag types provided (C-VLAN or S-VLAN). The VLAN tag Insert/Replace control bits for both tags is applicable to all segments. If the Double VLAN feature is not selected, then the application must not set the TSE bit in TDES3 for a TCP/IP packet with two tags. The DMA behavior in this scenario is unpredictable.

If the TSE bit is set in TDES3 for the packet and TCP header length provided is less than 5 (meaning this is an invalid TCP header because it is less than 20 bytes), the DMA does not perform segmentation, instead it transmits the entire packet as a single packet. In this scenario, the CRC pad control bits are forced by DMA to 00 (MAC does CRC and padding) and checksum insertion control bits are forced to 11 (hardware does the checksum calculation for both header and payload).

57.5.7 IPv4 ARP offload

The MAC supports the Address Recognition Protocol (ARP) Offload for IPv4 packets. This feature allows to process the IPv4 ARP request packet in the receive path and to generate the corresponding ARP response packet in the transmit path.

The MAC generates the ARP reply packets for appropriate ARP request packets. ARP packets for IPv4 are L2 layer packets with Length/Type of 0x0806.

The ARP offloading sequence is as follows:
1. The MAC receiver gets an ARP request if the request Target Protocol Address matches the IPv4 address programmed in the MAC L3 register.
2. The MAC generates an ARP reply packet.
3. The MAC copies the Sender Hardware Address field in the ARP request to the following fields:
   - DA field of the Ethernet packet header
   - Target Hardware Address field of the ARP reply packet
4. The MAC copies the Sender Protocol Address field in the ARP request to the Target Protocol Address field in the ARP reply packet.
5. The MAC places its MAC address in the following fields:
   - SA field of the Ethernet packet header
   - Sender Hardware Address field of the ARP reply packet
6. The MAC copies the Target Protocol Address field in the ARP request to the Sender Protocol Address field in the ARP reply packet.
7. The MAC sets the opcode field in ARP reply packet to 2 indicating ARP reply.
8. The MAC recalculates the CRC and performs padding for the generated ARP reply packet.
9. The MAC transmitter sends the ARP reply

The MAC processes only one ARP request at a time. It does not store the fields of multiple ARP requests. If the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC does not generate the ARP reply for new ARP request. The MAC forwards the new ARP request packet to the application with ARP Reply Not Generated status bit set (bit 34). However, in power-down mode, if the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC drops the new ARP request. If the Disable CRC check (DCRCC) bit of the Extended operating mode configuration register (ETH_MACECR) is set, then the MAC does not check for valid CRC of a ARP
request packet. It can generate an ARP response packet if the other conditions are valid. The ARP request packet must always have a valid CRC.

Note: When the received ARP request is less than the 64-byte packet length, the MAC does not send an ARP response. It is treated as a normal packet and forwarded to the application based on the MAC filter settings.

57.5.8 Loopback

The MAC supports loopback of transmitted packets to its receiver.

Guidelines for using Loopback mode

Below some guidelines for using the Loopback mode:

- Enable loopback only with the Full-duplex mode. In Half-duplex mode, the carrier sense signal or collision signal inputs get sampled which may result into issues such as packet dropping.
- If the Loopback mode is enabled without connecting a PHY chip, externally generate the Tx and Rx clocks and provide these clocks to the MAC.
- Do not loop back big packets since they may get corrupted in the loopback FIFO.

The Transmit and Receive clocks can have an asynchronous timing relationship. Therefore, an asynchronous FIFO is used to make the loopback path of the transmitted data to the Receive path. The FIFO is free-running to write on the write clock (eth_mii_tx_clk) and read on every read clock (eth_mii_rx_clk). At the start of each packet read from the FIFO, the write and read pointers are reinitialized to have an offset of four (in 10/100 Mbps mode). This avoids overflow or underflow during a packet transfer, and ensures that the overflow or underflow occurs only during the IPG period between the packets. The FIFO depth of five or nine is sufficient to prevent data corruption for packet sizes up to 9,022 bytes with a difference of 200 ppm between MII Transmit and Receive clock frequencies.

Therefore, bigger packets should not be looped back because they may get corrupted in this loopback FIFO.

At the end of every received packet, the Receive Protocol Engine module generates received packet status and sends it to the Receive Packet Controller module. The control, missed packet, and filter fail status are added to the Receive status in the Receive Packet Controller module. The MAC does not process ARP or PMT packets that are looped back.

Enabling Loopback mode

To enable this feature, program the LM bit of the Operating mode configuration register (ETH_MACCR). Loopback can be enabled for all PHY interfaces. The data is always looped back through internal asynchronous FIFO on to the internal Receive MII interface, irrespective of which PHY interface is selected.

The loopback data is also passed through the corresponding transmit PHY interface block, onto the Ethernet line.

Note: During loopback, the data/packet is reflected on mii_txd signal. Preemption is not supported in Loopback mode.
57.5.9 Flow control

The transmit flow control involves transmitting Pause packets in Full-duplex mode and back-pressure in Half-duplex mode to control the flow of packets from the remote end. This section describes the flow control for transmit and receive paths.

Flow control in Full-duplex mode

In Full-duplex mode, the MAC uses IEEE 802.3x Pause packets for flow control. Table 642 describes the fields of a Pause packet.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA</td>
<td>Contains the special multicast address</td>
</tr>
<tr>
<td>SA</td>
<td>Contains the MAC address 0</td>
</tr>
<tr>
<td>Type</td>
<td>Contains 8008</td>
</tr>
<tr>
<td>MAC Control opcode</td>
<td>Contains 0001 for IEEE 802.3x Pause Control packets</td>
</tr>
<tr>
<td>PT</td>
<td>Contains Pause time specified in the PT field of the Tx Queue flow control register (ETH_MACQTXFCR)</td>
</tr>
</tbody>
</table>

When the FCB bit is set, the MAC generates and transmits a single Pause packet. If the FCB bit is set again after the Pause packet transmission is complete, the MAC sends another Pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously-transmitted Pause packet, the application should program the Pause Time register with appropriate value and then again set the FCB bit.

Flow control in Half-duplex mode

In Half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

Table 643 describes the flow control in the Tx path based on the setting of the following bits:
- TFE bit of Tx Queue flow control register (ETH_MACQTXFCR)
- DM bit of Operating mode configuration register (ETH_MACCR)

Flow control is similar for all queues.
Transmit flow control

The transmit flow control is enabled when TFE bit is set in Tx Queue flow control register (ETH_MACQTXFCR).

Flow control trigger

The application can request the MAC to send a Pause packet or initiate back-pressure by setting the FCB bit in the corresponding Tx Queue flow control register (ETH_MACQTXFCR).

Receive flow control

In the Receive path, the flow control is functional only in Full-duplex mode. If any Pause packet is received in Half-duplex mode, the packet is considered as a normal control packet.

Description of receive flow control

Table 644 describes the flow control in the Rx path based on the setting of the following bits:
- RFE bit of Rx flow control register (ETH_MACRXFCR)
- DM bit of Operating mode configuration register (ETH_MACCR)

### Table 643. Tx MAC flow control

<table>
<thead>
<tr>
<th>TFE</th>
<th>DM</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>The MAC transmitter does not perform the flow control or backpressure operation.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>The MAC transmitter performs backpressure when Bit 0 of Tx Queue flow control register (ETH_MACQTXFCR) is set.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>The MAC transmitter sends the Pause packet when Bit 0 of Tx Queue flow control register (ETH_MACQTXFCR) is set.</td>
</tr>
</tbody>
</table>

### Table 644. Rx MAC flow control

<table>
<thead>
<tr>
<th>RFE</th>
<th>DM</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>The MAC receiver does not detect the received Pause packets.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>The MAC receiver does not detect the received Pause packets but recognizes such packets as Control packets.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>The MAC receiver detects or processes the Pause packets and responds to such packets by stopping the MAC transmitter.</td>
</tr>
</tbody>
</table>

The following sequence describes the Rx flow control:

1. The MAC checks the destination address (DA) of the received Pause packet for either of the following:
   - Multicast destination address: the DA matches the unique multicast address specified for the control packet (0x0180 C200 0001).
   - Unicast destination address: the DA matches the content of the MAC Address 0 registers (MAC Address 0 high register (ETH_MACA0HR) and MAC Address x...


low register (ETH_MACAxLR)) and the UP bit of Rx flow control register (ETH_MACRXFCR) is set.

If the UP bit is set, the MAC processes Pause packets with unicast destination address in addition to the unique multicast address.

2. The MAC decodes the following fields of the received packet:
   – Type field: this field is checked for 0x8808.
   – Opcode field: this field is checked for 0x0001 (Pause packet).
   – Pause time: the Pause time (for Pause packet) is captured to determine the time for which the transmitter needs to be blocked.

3. If the byte count of the status indicates 64 bytes and there is no CRC error, the MAC transmitter pauses the transmission of any data packet for the duration of the decoded Pause Time value multiplied by the slot time (64 byte times).

If subsequent Pause packets are received before the earlier Pause Time expires, the MAC updates the Pause Timer with new value.

**Enabling receive flow control**

Set the RFE bit in the Rx flow control register (ETH_MACRXFCR) to enable the Pause flow control.
57.5.10 MAC management counters

The peripheral supports storing the statistics about the received and transmitted packets in registers that are accessible through the application.

The counters in the MAC management counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted packets. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the AHB slave interface in the same way the CSR registers are accessed. The organization of these registers is shown in Section 57.11.4: Ethernet MAC and MMC registers.

The MMC counters are free running. There is no separate enable for the counters to start. A particular MMC counter starts counting when corresponding packet is received or transmitted.

The Receive MMC counters are updated for packets that are passed by the Address Filter (AFM) block. The statistics of packets dropped by the AFM module, are not updated unless they are runt packets of less than 6 bytes (DA bytes are not received fully). To get statistics of all packets, set bit 0 in the Packet filtering control register (ETH_MACPFR). The MMC module gathers statistics on encapsulated IPv4, IPv6, TCP, UDP, or ICMP payloads in received Ethernet packets.

In addition to control registers, two sets of registers are implemented:

- 6 registers used for collision, error and good packets counters:
  - Tx single collision good packets register (Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS))
  - Tx multiple collision good packets register (Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS))
  - Tx packet count good register (Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD))
  - Rx CRC error packets register (Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS))
  - Rx alignment error packets register (Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS))
  - Rx unicast packets good register (Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD))

- 4 registers to record LPI mode transition:
  - Tx LPI microsecond timer register (Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR))
  - Tx LPI transition counter register (Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR))
  - Rx LPI microsecond counter register (Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR))
  - Rx LPI transition counter register (Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR))
Definitions

The following terminology is used in MMC register descriptions:

- Transmitted packets are considered “good” if transmitted successfully. In other words, a transmitted packet is good if the packets transmission is not aborted because of any of the following errors:
  - Jabber timeout
  - No carrier or loss of carrier
  - Late collision
  - Packet underflow
  - Excessive deferral
  - Excessive collision

- Received packets are considered “good” if none of the following errors exists:
  - CRC error
  - Runt packet (shorter than 64 bytes)
  - Alignment error (in 10/100 Mbps only)
  - Length error (non-Type packet only)
  - Out of range (non-Type packet only, longer than 1518 bytes)

- The maximum transmit frame size depends on the frame type, as follows:
  - Untagged frame maxsize = 1,518
  - VLAN frame maxsize = 1,522
  - Jumbo frame maxsize = 9,018
  - JumboVLAN frame maxsize = 9,022

- The maximum receive packet size depends on the packet type and control bits (JE, S2KP, GPSLCE and EDVLP), as shown in the Table 645.

<table>
<thead>
<tr>
<th>JE</th>
<th>S2KP</th>
<th>GPSLCE</th>
<th>EDVLP</th>
<th>Untagged frame maximum size in bytes</th>
<th>Single VLAN frame maximum size in bytes</th>
<th>Double VLAN Frame maximum size in bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>9018</td>
<td>9022</td>
<td>9026</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>2000</td>
<td>2000</td>
<td>2000</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>GPSL</td>
<td>GPSL+4</td>
<td>GPSL+8</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1518</td>
<td>1522</td>
<td>1526</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>9018</td>
<td>9022</td>
<td>9022</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>GPSL</td>
<td>GPSL+4</td>
<td>GPSL+4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1518</td>
<td>1522</td>
<td>1522</td>
</tr>
</tbody>
</table>
57.5.11 Interrupts generated by the MAC

Interrupts can be generated from the MAC as a result of various events. These interrupt events are combined with the events in the DMA on the eth_sbd_intr_it signal. The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The *Interrupt status register (ETH_MACISR)* describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt on the eth_sbd_intr_it signals by setting the corresponding mask bits in the *Interrupt enable register (ETH_MACIER)*.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

57.5.12 MAC and MMC register descriptions

Refer to *Section 57.11.4: Ethernet MAC and MMC registers.*
57.6 **Ethernet functional description: PHY interfaces**

The Ethernet peripheral support several PHY interfaces. The root interface is the MII one. All other interfaces are derived from it as shown in Figure 806.

![Supported PHY interfaces](image)

This section describes the SMA module used for PHY control and different PHY interfaces. It contains the following sections:

- **Station management agent (SMA)**
- **Media independent interface (MII)**
- **Reduced media independent interface (RMII)**

57.6.1 **Station management agent (SMA)**

The application can access the PHY registers through the station management agent (SMA) module. The SMA includes a two-wire station management interface (MIM).

The SMA module supports accessing up to 32 PHYs. The application can address one of the 32 registers from any 32 PHYs. Only one register in one PHY can be addressed at a time. The application sends the control data to the PHY and receives status information from the PHY through the SMA module, as shown in Figure 807.

![SMA Interface block](image)

### SMA functional overview

The MAC initiates the management write or read operation with respect to the MDC clock. The MDC clock is derived from the CSR clock (eth_hclk). The maximum operating frequency of the ETH_MDC pin is 2.5 MHz, as specified in IEEE 802.3 specifications. However, a different divider can be selected if the system supports higher clock frequencies. The division factor depends on the clock range setting through CR[3:0] in the MDIO address register (ETH_MACMDIOAR) register. The MDC clock is selected as follows:

### Table 646. MCD clock selection

<table>
<thead>
<tr>
<th>Selection</th>
<th>eth_hclk</th>
<th>MDC clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>60–100 MHz</td>
<td>CSR clock/42</td>
</tr>
<tr>
<td>0001</td>
<td>100–150 MHz</td>
<td>CSR clock/62</td>
</tr>
<tr>
<td>0010</td>
<td>20–35 MHz</td>
<td>CSR clock/16</td>
</tr>
<tr>
<td>0011</td>
<td>35–60 MHz</td>
<td>CSR clock/26</td>
</tr>
</tbody>
</table>
The data exchange between the MAC and the PHY is performed through a three-state buffer and brought out as ETH_MDIO line connected to the PHY.

*Figure 808* shows the structure of a Clause 45 MDIO packet, while *Table 647* provides a detailed description of the packet fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLE</td>
<td>The ETH_MDIO line is three-state; there is no clock on ETH_MDC.</td>
</tr>
<tr>
<td>PREAMBLE</td>
<td>32 continuous bits of value 1</td>
</tr>
<tr>
<td>START</td>
<td>Start of packet is 0b00</td>
</tr>
</tbody>
</table>
| OPCODE  | – 0b00: Address  
– 0b01: Write  
– 0b10: Read+ Address  
– 0b11: Read                                                                 |
| PHY ADDR| 5-bit address select for one of 32 PHYs                                                                                                 |
| DEV TYPE| 5-bit device type                                                                                                                        |
| TA      | Turnaround  
– 0bZ0: Read and post-read increment address  
– 0b10: Write and address MDIO accesses where Z is the tri-state level                                                                 |
| DATA    | 16-bit value: for an address cycle (OPCODE = 0b00), this frame contains the address of the register to be accessed on the next cycle. The data cycle of a write frame, this field contains the data to be written to the register. For read or post-read increment address frames, this field contains the contents of the register read from the PHY.  
– In address and data write cycles, the peripheral drives the ETH_MDIO line during the transfer of these 16 bits.  
– In read and post-read increment address cycles, the PHY drives the ETH_MDIO line during the transfer of these 16 bits. |
The frame structure for Clause 22 frames is also supported. The C45E bit in MDIO address register (ETH_MACMDIOAR) can be programmed to enable Clause 22 or Clause 45 mode of operation. Figure 809 shows the structure of a Clause 22 MDIO packet, while Table 648 provides a detailed description of the packet fields.

Figure 809. MDIO packet structure (Clause 22)

<table>
<thead>
<tr>
<th>IDLE</th>
<th>PREAMBLE</th>
<th>START</th>
<th>OPCODE</th>
<th>PHY ADDR</th>
<th>REG ADDR</th>
<th>TA</th>
<th>DATA</th>
</tr>
</thead>
</table>

Table 648. MDIO Clause 22 frame structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLE</td>
<td>The ETH_MDIO line is three-state; there is no clock on ETH_MDC.</td>
</tr>
<tr>
<td>PREAMBLE</td>
<td>32 continuous bits of value 1</td>
</tr>
<tr>
<td>START</td>
<td>Start of packet is 0b01</td>
</tr>
<tr>
<td>OPCODE</td>
<td>0b10 for Read and 0b01 for Write</td>
</tr>
<tr>
<td>PHY ADDR</td>
<td>5-bit address select for one of 32 PHYs</td>
</tr>
<tr>
<td>REG ADDR</td>
<td>Register address in the selected PHY</td>
</tr>
<tr>
<td>TA</td>
<td>Turnaround</td>
</tr>
<tr>
<td></td>
<td>– 0bZ0: Read and post-read increment address</td>
</tr>
<tr>
<td></td>
<td>– 0b10: Write and address MDIO accesses where Z is the tri-state level</td>
</tr>
<tr>
<td>DATA</td>
<td>Any 16-bit value. In a Write operation, the MAC drives ETH_MDIO. In a Read operation, the PHY drives it.</td>
</tr>
</tbody>
</table>

In addition to normal read and write operations, the SMA also supports post-read increment address while operating in Clause 45 mode.

MII management write operations

After the station management agent receives the PHY address and the write data from the MAC CSR module, the SMA starts a write operation to the PHY registers.

Figure 810 illustrates the flow for a write operation from the SMA module to the PHY registers.
When bits[3:2] are set to 01 and bit 0 to 1 in the MDIO address register (ETH_MACMDIOAR), the MAC CSR module transfers the PHY address, the register address in PHY, and the write data (MDIO data register (ETH_MACMDIODR)) to the SMA to initiate a Write operation into the PHY registers. At this point, the SMA module starts a Write operation on the MII management interface using the management packet format specified in the MII specifications (as per IEEE 802.3-2002 specifications, Section 22.2.4.5).

When the SMA module starts a Write operation, the write data packet is transmitted on the MDIO line. The MAC drives the MDIO line for complete duration of the packet. The Busy bit is set high until the write operation is complete. The CSR ignores the Write operations performed to the MDIO address register (ETH_MACMDIOAR) or the MDIO data register.
(ETH_MACMDIODR) during this period (the Busy bit is high). When the Write operation is complete, the SMA module indicates this to the CSR, and the CSR resets the Busy bit. The packet format for the Write operation is as follows:

Figure 811. Write data packet

<table>
<thead>
<tr>
<th>IDLE</th>
<th>PREAMBLE</th>
<th>START</th>
<th>OPCODE</th>
<th>PHY ADDR</th>
<th>REG ADDR</th>
<th>TA</th>
<th>DATA</th>
<th>IDLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>1111...11</td>
<td>01</td>
<td>01</td>
<td>AAAAA</td>
<td>RRRRR</td>
<td>10</td>
<td>00</td>
<td>Z</td>
</tr>
</tbody>
</table>

**MII management read operation**

When bits[3:2] are set to 11 and bit 0 to 1 in the MDIO address register (ETH_MACMDIOAR), the MAC CSR module transfers the PHY address and the register address in PHY to the SMA to initiate a Read operation in the PHY registers. At this point, the SMA module starts a Read operation on the MII management interface using the management packet format specified in the MII specifications (as per IEEE 802.3-2002 specifications, Section 22.2.4.5).

When the SMA module starts a Read operation on the MDIO, the CSR ignores the Write operations to the MDIO address register (ETH_MACMDIOAR) or MDIO data register (ETH_MACMDIODR) during this period (the Busy bit is high) and the transaction is completed without any error on the MCI interface. When the Read operation is complete, the SMA indicates this to the CSR. The CSR resets the Busy bit and updates the MDIO data register (ETH_MACMDIODR) with the data read from the PHY. The MAC drives the MDIO line for the complete duration of the frame except during the Data fields when the PHY is driving the MDIO line. For more information about the communication from the application to the PHYs, see the Reconciliation Sublayer and Media Independent Interface Specifications sections of the IEEE 802.3z, 1000BASE Ethernet.

The packet format for the Read operation is as follows:

Figure 812. Read data packet

<table>
<thead>
<tr>
<th>IDLE</th>
<th>PREAMBLE</th>
<th>START</th>
<th>OPCODE</th>
<th>PHY ADDR</th>
<th>REG ADDR</th>
<th>TA</th>
<th>DATA</th>
<th>IDLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>1111...11</td>
<td>01</td>
<td>10</td>
<td>AAAAA</td>
<td>RRRRR</td>
<td>Z0</td>
<td>00</td>
<td>Z</td>
</tr>
</tbody>
</table>
**Preamble suppression**

The IEEE standard specifies 32-bit preamble (all-ones) for the MDIO frames. The peripheral provides controls to support preamble suppression. It transmits MDIO frames with only 1 preamble bit. The preamble suppression can be enabled by setting the PSE bit in *MDIO address register (ETH_MACMDIOAR)*.

**Trailing clocks and back-to-back transactions**

The peripheral drives the ETH_MDC clock for the duration of the MDIO frame. There is no clock driven during the idle period. The trailing clock feature can be used if the PHY needs the ETH_MDC clock to be active for some cycles after the MDIO frame. The NTC[2:0] bitfield in *MDIO address register (ETH_MACMDIOAR)* allows the programming of trailing clocks from 0 to 7.

The peripheral supports back-to-back transactions which allow starting the next MDIO frame even before the trailing clocks are complete for previous MDIO frame. This feature can be enabled by setting BTB bit in *MDIO address register (ETH_MACMDIOAR)* when the trailing clock feature is also enabled. When back-to-back transactions are enabled, the GMII busy bit (GB) is cleared immediately after MDIO frame completion. This allows the software to issue the next command, which is executed by the peripheral while trailing clocks are still on for the previous MDIO frame. When (GB) transactions are not enabled, the GMII busy bit is cleared after the trailing clocks are complete for the MDIO frame.

**Interrupt for MDIO transaction completion**

The peripheral can generate an interrupt on completion of MDIO read or write transactions. Therefore, the application need not poll the GMII busy bit of the *MDIO address register (ETH_MACMDIOAR)* to know the completion of MDIO commands.
57.6.2 **Media independent interface (MII)**

The media-independent interface (MII) defines the interconnection between the MAC sublayer and the PHY for data transfer at 10 Mbit/s and 100 Mbit/s.

MII signals are given in *Figure 813: Media independent interface (MII) signals*.

*Figure 813. Media independent interface (MII) signals*

- **TX_CLK**: continuous clock that provides the timing reference for Tx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s and 25 MHz at 100 Mbit/s.
- **TXD[3:0]**: transmit data.
  TXD is a bundle of 4 data signals driven synchronously by the MAC sublayer and qualified (valid data) on the assertion of the TX_EN signal. TXD[0] is the least significant bit, TXD[3] is the most significant bit. While TX_EN is deasserted, the transmit data must have no effect upon the PHY.
- **TX_EN**: transmission enable signal indicating that the MAC is presenting nibbles on the MII for transmission. It must be asserted synchronously (TX_CLK) with the first nibble of the preamble and must remain asserted while all nibbles to be transmitted are presented to the MII.
- **TX_ER (optional)**: required only for Energy Efficient Ethernet (EEE). The transmit error is indicated by inverting the CRC. The remote station can detect the Transmit error through incorrect CRC.
- **RX_CLK**: continuous clock that provides the timing reference for Rx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s, 25 MHz at 100 Mbit/s.
- **RXD[3:0]**: receive data
  RXD is a bundle of 4 data signals driven synchronously by the PHY and qualified (valid data) on the assertion of the RX_DV signal. RXD[0] is the least significant bit, RXD[3] is...
the most significant bit. While RX_EN is deasserted and RX_ER is asserted, a specific RXD[3:0] value is used to transfer specific information from the PHY.

- **RX_DV**: receive data valid
  This signal indicates that the PHY is presenting recovered and decoded nibbles on the MII for reception. It must be asserted synchronously (RX_CLK) with the first recovered nibble of the frame and must remain asserted through the final recovered nibble. It must be deasserted prior to the first clock cycle that follows the final nibble. In order to receive the frame correctly, the RX_DV signal must encompass the frame, starting no later than the SFD field.

- **RX_ER**: receive error
  This signal must be asserted for one or more clock periods (RX_CLK) to indicate to the MAC sublayer that an error was detected somewhere in the frame. This error condition must be qualified by RX_DV assertion.

- **CRS**: carrier sense.
  This signal is asserted by the PHY when either the transmit or receive medium is non idle. It is deasserted by the PHY when both transmit and receive media are idle. The PHY must ensure that the CS signal remains asserted throughout the duration of a collision condition. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode the state of this signal is don’t care for the MAC sublayer.

- **COL**: collision detection signal
  This signal must be asserted by the PHY upon detection of a collision on the medium and must remain asserted while the collision condition persists. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode, the state of this signal is don’t care for the MAC sublayer.

### 57.6.3 Reduced media independent interface (RMII)

The reduced media independent interface (RMII) specification reduces the pin count between Ethernet PHYs and STM32 MCU. According to the IEEE 802.3u, an MII contains 16 pins for data and control. RMII specification reduces the pin count to 7.

Part of the Ethernet peripheral, the RMII module is instantiated at the MAC output. This helps in translating the MII of the MAC into the RMII. The RMII block has the following characteristics:

- Supports 10 Mbps and 100 Mbps operating rates. It does not support the 1000 Mbps operation.
- Provides independent 2-bits wide Transmit and Receive paths by sourcing two clock references externally.
**RMII block diagram**

*Figure 814: RMII block diagram* shows the position of the RMII block relative to the MAC and RMII PHY. The RMII block is placed in front of the MAC to translate the MII signals to RMII signals.

**Figure 814. RMII block diagram**

- **RMII_REF_CLK**: continuous 50 MHz reference clock input
- **TXD[1:0]**: transmit data
- **TX_EN**: transmit data enable. When high, this bit indicates that valid data are being transmitted on TXD[1:0].
- **RXD[1:0]**: receive data
- **CRS_DV**: carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.
Transmit bit order

Each nibble from the MII interface must be transmitted on the RMII interface di-bit at a time with the order of di-bit transmission shown in *Figure 815: Transmission bit order*. The lower order bits (D1 and D0) are transmitted first followed by higher order bits (D2 and D3).

*Figure 815. Transmission bit order*
Receive bit order

Each nibble is transmitted to the MII interface from the di-bit received from the RMII interface in the nibble transmission order shown in Figure 816: Receive bit order. The lower order bits (D0 and D1) are received first, followed by the higher order bits (D2 and D3).

Figure 816. Receive bit order
57.7 Ethernet low-power modes

57.7.1 Low-power management

The Ethernet peripheral supports the following techniques to save power:

- Magic packet
- Remote wake-up

The magic packet and remove wake-up techniques are used to save power in the host system when it is idle (Sleep mode) and has to be woken up only at the reception of specific packets from the Ethernet network. In Sleep mode, the power to the host logic along with the majority of the peripheral (except the MAC receiver logic), can be shut down. On receiving the specific packets from the network, the MAC provides the trigger to restore the power to the host system and come back to normal state.

The Energy Efficient Ethernet (EEE) mode is compliant with the IEEE 802.3az-2010 standard. It is primarily targeted to save power in the Ethernet port when there is no traffic on the line. In this mode, the host indicates to the far-end that it does not have any packets to transmit in the near future and puts the transmitter port (MAC controller, PCS and PHY layers) in low-power mode. Similarly, the receiver port can also be put in low-power mode when the far-end host indicates that it does not have any traffic to transfer. This allows significant saving of power in the Ethernet port (mainly in the PHY) with intermittent and bursty traffic profile. The triggering of entry and exit out of the EEE mode is controlled by the MAC and is supported within the peripheral.

Simultaneous operation of the EEE mode along with any or both the other power saving modes is also supported.

Description of magic packet mode

This section describes how to save power through magic packet detection.

Note: The magic packet feature is based on the magic packet technology white paper from Advanced Micro Device (AMD).

The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit in Operating mode configuration register (ETH_MACCR) and PWE bit in Watchdog timeout register (ETH_MACWTR).

In the magic-packet-based power saving mode, the reception of a valid magic packet by the MAC receiver triggers an exit from low-power mode. The MAC enters power saving mode when PWRDWN bit of PMT control status register (ETH_MACPCSR) is programmed to 1.

Exit from the magic-packet-based power saving mode is enabled by setting the MGKPKTEN bit of PMT control status register (ETH_MACPCSR) to 1.

The magic packet contains a unique pattern at any offset after the Destination address, Source address, and Length/Type fields. In addition to the unique pattern matching, the MAC receiver also checks for the following, to detect the received packet as a valid magic packet:

- The packet must be addressed to it (Destination Address of the received packet should perfect match the MAC Address 0 high register (ETH_MACA0HR) and MAC Address 0 low register (ETH_MACA0LR)) or with multicast/broadcast address.
- The packet must not have any length error, FCS error, dribble bit error, GMII error, and collision.
- The packet must not be runt (length including Ethernet header and FCS is at least 64 bytes).

**Magic packet data format**

The content of the unique pattern in magic packets is described as follows:

- Six bytes of all-ones (0xFF FF FF FF FF FF) called synchronization stream. There can be more than six bytes of 0xFF, but only the last six are considered.
- The synchronization stream is immediately followed by 16 repetitions of the Destination address field of the packet (MAC Address 0 high register (ETH_MACA0HR) and the MAC Address 0 low register (ETH_MACA0LR)) or multicast/broadcast address.
- No break or interruption between synchronization stream and first repetition of Destination address field or within its 16 repetitions.

If the MAC address of a node is 0x00 11 22 33 44 55, the MAC scans for the following data sequence:

Destination Address Source Address Length/Type............... FF FF FF FF FF FF
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
...CRC

**Description of remote wake-up packet mode**

This section describes the power saving mode based on remote wake-up packet.

*Note:* The remote wake-up packet feature implementation is based on the Device Class Power Management Reference Specification and various implementation-specific white papers.

The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit in Operating mode configuration register (ETH_MACCR) and PWE bit in Watchdog timeout register (ETH_MACWTR).

In the remote-wake-up-magic-packet-based power saving mode, the reception of expected remote wake-up packet by the MAC receiver triggers the exit from low-power mode. The MAC enters power saving mode when PWRDWN bit in PMT control status register (ETH_MACPCSR) is programmed to 1. Exit from the remote-wake-up-magic-packet-based power saving mode is enabled by programming RWKPKTEN bit of PMT control status register (ETH_MACPCSR) to 1.

The MAC implements a filter lookup table (programmed through Remote wake-up packet filter register (ETH_MACRWKPFR) in which CRC, offset, and byte mask of the pattern embedded in remote wake-up packet and the filter operation commands are programmed.

The pattern embedded in the remote wake-up packet is located at any offset after the Destination address and Source address fields. In addition to the CRC match for the pattern, the MAC receiver also checks the following, to detect the received packet as a valid remote wake-up packet:

- The packet must be addressed to it (Destination Address of the received packet should perfect match the MAC Address 0 high register (ETH_MACA0HR) and MAC Address 0 low register (ETH_MACA0LR)) or with multicast/broadcast address.
- The packet must not have any length error, FCS error, dribble bit error, GMII error, and collision.
• The packet must not be runt (length including Ethernet header and FCS is at least 64 bytes).

When a valid remote wake-up packet is received, the MAC receiver sets the RWKPRCVD bit in PMT control status register (ETH_MACPCSR) and triggers the interrupt on pmt_intr_o output port. The PMTIS bit in Interrupt status register (ETH_MACISR) is set when power-gating is not enabled in low-power mode. An interrupt is triggered to the application on the sbd_intr_o output port when interrupt is enabled (PMTIE bit in Interrupt enable register (ETH_MACIER) is set) and CSR clock is not gated off in low-power mode.

Remote wake-up packet filters

When the remote-wake-up-based power saving mode is enabled, four remote wake-up filters can be selected. The structure of the remote wake-up filters is shown in Table 649: Remote wake-up packet filter register.

Table 649. Remote wake-up packet filter register

<table>
<thead>
<tr>
<th>ETH_MACRWKPFR value</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Filter 0 byte mask</td>
</tr>
<tr>
<td>1</td>
<td>Filter 1 byte mask</td>
</tr>
<tr>
<td>2</td>
<td>Filter 2 byte mask</td>
</tr>
<tr>
<td>3</td>
<td>Filter 3 byte mask</td>
</tr>
<tr>
<td>4</td>
<td>Reserved Filter 3 command</td>
</tr>
<tr>
<td>5</td>
<td>Filter 3 offset</td>
</tr>
<tr>
<td>6</td>
<td>Filter 1 CRC - 16</td>
</tr>
<tr>
<td>7</td>
<td>Filter 3 CRC - 16</td>
</tr>
</tbody>
</table>

The remote wake-up filter fields are described in Table 650: Description of the remote wake-up filter fields.
Table 650. Description of the remote wake-up filter fields

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter i Byte mask</td>
<td>The filter i byte mask register defines the bytes of the packet that are examined by filter i (0, 1, 2 or 3) to determine whether or not a packet is a wake-up packet.</td>
</tr>
<tr>
<td></td>
<td>– The MSB (31st bit) must be zero.</td>
</tr>
<tr>
<td></td>
<td>– Bit j[30:0] is the byte mask.</td>
</tr>
<tr>
<td></td>
<td>– If Bit j (byte number) of the byte mask is set, the CRC block processes the Filter i Offset + j of the incoming packet; otherwise Filter i Offset + j is ignored.</td>
</tr>
<tr>
<td>Filter i Command</td>
<td>The 4-bit filter i command controls the filter i operation.</td>
</tr>
<tr>
<td></td>
<td>– Bit 3 specifies the address type, defining the destination address type of the pattern.</td>
</tr>
<tr>
<td></td>
<td>– When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packets.</td>
</tr>
<tr>
<td></td>
<td>– Bit 2 (Inverse mode), when set, reverses the logic of the CRC16 Hash function signal, to reject a packet with matching CRC_16 value.</td>
</tr>
<tr>
<td></td>
<td>– Bit 2, along with Bit 1, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as “Pattern 1 AND NOT Pattern 2”.</td>
</tr>
<tr>
<td></td>
<td>– Bit 1 (And_Previous) implements the Boolean logic(1).</td>
</tr>
<tr>
<td></td>
<td>– When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set.</td>
</tr>
<tr>
<td></td>
<td>– Bit 0 is the enable for filter i. If Bit 0 is not set, filter i is disabled.</td>
</tr>
<tr>
<td>Filter i Offset</td>
<td>This filter i offset register defines the offset (within the packet) from which the filter i examines the packets.</td>
</tr>
<tr>
<td></td>
<td>– This 8-bit pattern-offset is the offset for the filter i first byte to be examined.</td>
</tr>
<tr>
<td></td>
<td>– The minimum allowed offset is 12, which refers to the 13th byte of the packet.</td>
</tr>
<tr>
<td></td>
<td>– The offset value 0 refers to the first byte of the packet.</td>
</tr>
<tr>
<td>Filter i CRC-16</td>
<td>This filter i CRC-16 register contains the CRC_16 value calculated from the pattern and also the byte mask programmed to the wake-up filter register block.</td>
</tr>
<tr>
<td></td>
<td>– The 16-bit CRC calculation uses the following polynomial: G(x) = x^16 + x^15 + x^2 + 1</td>
</tr>
<tr>
<td></td>
<td>– Each mask, used in the Hash function calculation, is compared with a 16-bit value associated with that mask. Each filter has the following:</td>
</tr>
<tr>
<td></td>
<td>– 32-bit Mask: Each bit in this mask corresponds to one byte in the detected packet. If the bit is 1, the corresponding byte is taken into the CRC16 calculation.</td>
</tr>
<tr>
<td></td>
<td>– 8-bit Offset Pointer: Specifies the byte to start the CRC16 computation. The pointer and the mask are used together to locate the bytes to be used in the CRC16 calculations.</td>
</tr>
</tbody>
</table>

1. The And_Previous bit setting is applicable within a set of four filters.

Setting And_Previous bit of a filter that is not enabled has no effect, that is setting And_Previous bit of lowest number filter in the set of four filters has no effect. For example, setting And_Previous bit of Filter 0 has no effect.

If And_Previous bit is set for a given filter to form an AND chained filter, the AND chain breaks when it finds a disabled filter. For example: If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set) but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result ANDed with Filter 3 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 2 is not enabled (bit 0 in Filter 2 command is reset), then since setting Filter 2 And_Previous bit has no effect, only Filter 1 result ORed with Filter 3 result is considered.

If filters chained by And_Previous bit setting have complementary programming, then a frame may never pass the AND chained filter. For example, if Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 1 Address_Type bit is set (bit 3 in Filter 1 command is set) indicating multicast detection and Filter 2 Address_Type bit is reset (bit 3 in Filter 2 command is reset) indicating unicast detection or vice versa, then a remote wake-up frame does not pass the AND chained filter as a remote wake-up frame cannot be of both unicast and multicast address types.
The remote wake-up filter registers are implemented as eight indirect access registers (wkuppktfilter_reg#i) for four remote wake-up filters, and accessed by the application through *Remote wake-up packet filter register (ETH_MACRWKPFR)*. The entire set of wkuppktfilter_reg registers must be written to program the remote wake-up filters. The wkuppktfilter_reg register is programmed by sequentially writing the eight register values in *Remote wake-up packet filter register (ETH_MACRWKPFR)* for wkuppktfilter_reg0 to wkuppktfilter_reg3, respectively. The wkuppktfilter_reg register is read in a similar way. The MAC updates the wkuppktfilter_reg register current pointer value in RWKPTR field of *PMT control status register (ETH_MACPCSR)*.

**Note:** If the *Remote wake-up packet filter register (ETH_MACRWKPFR)* is accessed in byte or half-word mode, the internal counter to access the appropriate wkuppktfilter_reg is incremented when the CPU accesses Lane 3.

*When* *Remote wake-up packet filter register (ETH_MACRWKPFR)* is written, the content is transferred from CSR clock domain to PHY receive clock domain after the write operation. There should not be any further write to the *Remote wake-up packet filter register (ETH_MACRWKPFR)* until the first write is updated in PHY receive clock domain. Otherwise, the second write operation does not get updated to the PHY receive clock domain. Therefore, the delay between two write operations to the *Remote wake-up packet filter register (ETH_MACRWKPFR)* should be at least 4 cycles of the PHY receive clock.

**PMT interrupt**

The PMT interrupt signal is asserted when a valid remote wake-up packet is received. *Table 651* lists the remote wake-up scenarios in which PMT interrupt is generated.

<table>
<thead>
<tr>
<th>Filter i Command</th>
<th>Frame Type and CRC Status</th>
<th>Interrupt Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAST</td>
<td>INV</td>
<td>EN</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. In all other combinations, the Remote wake-up packet is not detected and PMT interrupt is not generated.

In addition to sbd_intr_o signal, the pmt_intr_o (synchronous to Rx clock) signal is asserted. The pmt_intr_o signal, synchronous to the Rx clock domain, is provided so that the application clock can be stopped by software when the MAC is in the power-down mode. It is ORed with lpi_intr_o signal (see *Section : LPI interrupt*) and tied to the EXTI peripheral (line 86).

As the pmt_intr_o signal is generated in the PHY Rx clock domain, it is not cleared immediately when the *PMT control status register (ETH_MACPCSR)* is read. This is
because the resultant clear signal has to cross to the PHY Rx clock domain, and then clear the interrupt source. This delay is at least 4 clock cycles of Rx clock and can be significant when the peripheral is operating in the 10 Mbps mode. When the application clears the PWRDWN bit in Remote wake-up packet filter register (ETH_MACRWKPFR), the MAC comes out of the power-down mode, but this event does not generate the PMT interrupt.

Power-down sequence

The software must perform the following tasks to initiate the power-down sequence:

- Disable the Transmit DMA (if applicable) by clearing the ST bit of the Channel transmit control register (ETH_DMACTXCR).
- Wait for any previous frame transmissions to complete. You can check this by reading TFCSTS[1:0] and TPESTS bits in Debug register (ETH_MACDR) and TXQSTS bit in Tx queue debug register (ETH_MTLTXQDR) of all MTL Tx Queues.
- Disable the MAC transmitter and MAC receiver by clearing TE and RE bits in Operating mode configuration register (ETH_MACCR).
- Wait till the Receive DMA empties all frames from the Rx FIFO. You can check this by reading PRXQ[13:0] in Rx queue debug register (ETH_MTLRXQDR) of all Rx Queues. If these bits are zero, it indicates that the Rx FIFO is empty.
- Configure the magic packet (MGPKTEN) and/or remote wake-up (RWKPKTEN) detection in the PMT control status register (ETH_MACPCSR).
- Set bit 31 (ARPEN) in the Operating mode configuration register (ETH_MACCR).
- Enable the MAC Receiver by setting RE bit and then set PWRDWN bit in the PMT control status register (ETH_MACPCSR) to initiate the power-down sequence in MAC.

Note: If the feature is enabled and the MAC Transmitter is in the LPI mode when it is put into the power-down mode, then the MII interface gets clamped to assert the LPI pattern. If the MAC Transmitter is not in the LPI mode when it is put into the power-down mode, the GMII or MII interface gets clamped to all-zero.

Power-up sequence

The MAC wakes up on receiving the magic packet or remote wake-up frame. The power-up sequence is as follows:

- The MAC asserts pmt_intr_o. When only clock-gating is employed in low-power mode, the pmt_intr_o signal can be used to start the clocks that were gated-off after entering low-power mode.
- The software performs the following tasks:
  - De-assert the pmt_intr_o by reading the PMT control status register (ETH_MACPCSR).
  - Perform a write operation (with reset values) to the PMT control status register (ETH_MACPCSR) and the Remote wake-up packet filter register (ETH_MACRWKPFR) so that the corresponding values in the always-on block gets synchronized. Otherwise, the values of these registers are different.
  - Perform write operations to the Operating mode configuration register (ETH_MACCR), MAC Address 0 high register (ETH_MACROHR) and MAC Address 0 low register (ETH_MACA0LR) to synchronize the values in the CSR module and the respective bits in the always-on block. Otherwise, the MAC receiver is on even though the Receive Enable bit is set to 0.
After completing these steps, the software must initialize all registers, enable the transmitter, and program the DMA (in DMA configurations) to resume the normal operation.

### 57.7.2 Energy Efficient Ethernet (EEE)

EEE is an operational mode that enables the IEEE 802.3 Media Access Control (MAC) sublayer along with a family of physical layers to operate in the Low-Power Idle (LPI) mode. The EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbps. The peripheral supports the IEEE 802.3az-2010 for EEE.

The LPI mode allows saving power by switching off the parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functionalities to save power during the periods of low-link utilization. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY.

The EEE specifies the capabilities negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

#### Transmit path functions

The transmit path functions include tasks that the MAC must perform to make the PHY enter the LPI state.

In the transmit path, the software must set the LPIEN bit of the **LPI control and status register (ETH_MACLCSR)** to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and starts transmitting the LPI pattern instead of the IDLE pattern if the link status has been up continuously for a period specified in the LPI LS TIMER LST[9:0] bitfield of **LPI timers control register (ETH_MACLTCR)**. The PHY Link Status PLS bit of the **LPI control and status register (ETH_MACLCSR)** indicates the link status of the PHY.

**Note:** The EEE feature is not supported when the MAC is configured to use the RMII.

According to the standard (IEEE 802.3az-2010), the PHY must not stop the TxCLK clock during the LPI state in the MII (10 or 100) mode.

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. De-asserts TX_EN.
2. Asserts TX_ER.
3. Sets TXD[3:0] to 0x1 (for 100 Mbps)
4. Updates the status (TLPIEN bit of **LPI control and status register (ETH_MACLCSR)**) and generates an interrupt.

**Note:** The MAC maintains the same state of the TX_EN, TX_ER, and TXD signals for the entire duration during which the PHY remains in the LPI state.
To bring the PHY out of the LPI state, that is when the software resets the LPIEN bit, the MAC performs the following tasks:

1. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern.
2. Starts the LPI TW TIMER:
   The MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programmed in the TWT field of the LPI timers control register (ETH_MACLTCR).
3. Updates the LPI exit status (TLPIEX bit of the LPI control and status register (ETH_MACLCSR)) and generates an interrupt.

Figure 817 shows the behavior of TX_EN, TX_ER, and TXD[3:0] signals during the LPI mode transitions.

**Note:** The MAC does not stop the TX_CLK clock. The application can stop this clock (as shown in Figure 817) if the PHY supports it and when the MAC sets the sbd_tx_clk_gating_ctrl_o signal to 1. The sbd_tx_clk_gating_ctrl_o signal is asserted after nine Tx Clock Cycles, one Pulse Synchronizer delay, and one CSR clock cycle. The assertion of the sbd_tx_clk_gating_ctrl_o signal depends on the LPITCSE bit of the LPI control and status register (ETH_MACLCSR) and can be done automatically as shown on Figure 818.

If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern and so the Tx Clock cannot be gated.

If the MAC is in the Tx LPI mode and the Tx clock is stopped, the application should not write to CSR registers that are synchronized to Tx clock domain.

If the MAC is in the LPI mode and the application issues a soft reset or hard reset, the MAC transmitter comes out of the LPI mode.

---

**Figure 817. LPI transitions (Transmit, 100 Mbds)**

---
Automated entry/exit from LPI mode in transmit path

The MAC transmitter can be programmed to enter and exit LPI Idle mode automatically based on whether it is Idle for a specific period of time or has a packet to transfer. These modes are enabled and controlled by the LPI control and status register (ETH_MACLCSR).

When LPITXA and LPIEN of LPI control and status register (ETH_MACLCSR) are set, the MAC transmitter enters LPI Idle state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter exits the LPI Idle state and clears the LPITXEN bit as soon as any of functions in the TX path (DMA, MTL or MAC) becomes non-idle due to initiation of a packet transfer.

In addition, when LPITE is also set, the MAC transmitter enters LPI Idle state only if the Transmit path remains in idle state (no activity) for the time period indicated by the value in LPI entry timer register (ETH_MACLETR). In this mode also, the MAC transmitter exits the LPI Idle state as soon as any of the functions becomes non-idle. However, the LPIEN bit is not cleared but remains active so that reentry to LPI Idle state is possible without any software intervention when the MAC becomes idle again.

When both LPITE and LPITXA bits are cleared, the application can directly control the entry and exit of LPI Idle state by programming the LPIEN bit.

Receive path Functions

The receive path functions include the tasks that the PHY and MAC must perform when the PHY receives signals from the link partner to exit the LPI state.

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts RX_ER.
2. The PHY sets RXD[3:0] to 0x1 (for 100 Mbps).
3. The PHY de-asserts RX_DV.
4. The MAC updates the RLPIEN bit of the LPI control and status register (ETH_MACLCSR) and immediately generates an interrupt.
Note: The PHY maintains the same state of the RX_ER, RXD, and RX_DV signals for the entire duration during which it remains in the LPI state.

If the LPI pattern is detected for a very short duration (that is, less than two cycles of Rx clock), the MAC does not enter the Rx LPI mode.

If the duration between the end of the current Rx LPI pattern and the start of the next Rx LPI pattern is very short (that is, less than two Rx clock cycles), then the MAC exits and enters again the Rx LPI mode. The MAC does not trigger the Rx LPI Exit and Entry interrupts.

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY de-asserts RX_ER and returns to a normal inter-packet state.
2. The MAC updates the RLPIEX bit of the LPI control and status register (ETH_MACLCSR) and generates an interrupt immediately. The sideband signal lpi_intr_o (synchronous to Rx clock) is also asserted.

Figure 819 shows the behavior of RX_ER, RX_DV, and RXD[3:0] signals during the LPI mode transitions.

Figure 819. LPI transitions (receive, 100 Mbps)

Note: If the RX_CLK_stoppable bit (in the PHY register written through MDIO) is asserted when the PHY is indicating LPI to the MAC, the PHY may halt the RX_CLK at any time more than nine clock cycles after the start of the LPI state as shown in Figure 819.

If the MAC is in LPI mode and the application issues a soft reset or hard reset, the MAC receiver exits from LPI mode during reset. If the LPI pattern is still received after the reset is de-asserted, the MAC receiver enters again the LPI state.

If the RX clock is stopped in the RX LPI mode, the application should not write to the CSR registers that are being synchronized to the RX clock domain.

When the PHY sends the LPI pattern, if EEE feature is enabled, the MAC automatically enters the LPI state. There is no software control to prevent the MAC from entering the LPI state.
LPI timers

The transmitter maintains the LPI LS TIMER, LPI TW TIMER, and LPI AUTO ENTRY TIMER timers.

The following LPI timers are loaded with the respective values from the LPI timers control register (ETH_MACLTCR) and LPI entry timer register (ETH_MACLETR):

- **LPI LS TIMER**
  The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up. This timer is cleared every time the link goes down. It starts to increment when the link is up again and continues to increment until the value of the timer becomes equal to the terminal count. Once the terminal count is reached, the timer remains at the same value as long as the link is up. The terminal count is the value programmed in the LST[9:0] bitfield in the LPI timers control register (ETH_MACLTCR). The LPI LS TIMER is 10-bit wide. The software can program up to 1023 milliseconds.

- **LPI TW TIMER**
  The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count should be programmed in Bit[15:0] of LPI timers control register (ETH_MACLTCR). The terminal count of the timer is the value of resolved Transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. After exiting the LPI mode, the MAC resumes its normal operation after the TW timer reaches the terminal count.
  The MAC supports the LPI TW TIMER in units of microsecond. The LPI TW TIMER is 16-bit wide. Therefore, the software can program up to 65535 micro seconds.

- **LPI AUTO ENTRY TIMER**
  This timer counts in steps of eight microseconds, the time for which the MAC transmit path has to remain in idle state (no activity), before the MAC Transmitter enters the LPI IDLE state and starts transmitting the LPI pattern. This timer is enabled when LPITE bit in LPI control and status register (ETH_MACLCSR) is set.

LPI interrupt

The MAC generates the LPI interrupt when the Tx or Rx side enters or exits the LPI state. The interrupt sbd_intr_o is asserted when the LPI interrupt status is set. The LPI interrupt can be cleared by reading the LPI control and status register (ETH_MACLCSR).

When the MAC exits the Rx LPI state, then in addition to the sbd_intr_o, the sideband signal lpi_intr_o (synchronous to Rx clock) is asserted. The lpi_intr_o signal can be used to trigger the external clock-gating circuitry to restore the application clock to the MAC. The lpi_intr_o signal, synchronous to the Rx clock domain, is provided so that the application clock can be stopped by software when the MAC is in the LPI state. It is ORed with pmt_intr_o signal (see Section : PMT interrupt) and tied to the EXTI peripheral (line 86).

The lpi_intr_o signal is generated in the Rx clock domain. It may not be cleared immediately after the LPI control and status register (ETH_MACLCSR) is read. This is because the clear signal, generated in CSR clock domain, has to cross the Rx clock domain, and then clear the interrupt source. This delay is at least four clock cycles of Rx clock and can be significant when the peripheral is operating in the 10 Mbps mode.

**Programming guidelines for Energy Efficient Ethernet**

For detailed guidelines on the programming guidelines, see Section 57.9.11: Programming guidelines for Energy Efficient Ethernet (EEE) on page 2774.
57.8 Ethernet interrupts

The Ethernet peripheral generates a single interrupt signal (eth_sbd_intr_it). This signal can be raised as a result of various events. These events are captured in status registers and interrupt enables are provided for each source of interrupt such that the interrupt signal is asserted for an event only when the corresponding interrupt enable is set.

The interrupt status and corresponding enable registers are organized in a hierarchical manner so that it is easier for software to traverse and identify the source of interrupt event quickly. When interrupt is asserted, the Interrupt status register (ETH_DMAISR) register is first level that indicates the major blocks for the interrupt event source. This register is read-only, and it contains bits corresponding to each DMA channel (TX & RX pair), the MTL, and the MAC. The software application must then read one (or more) of the following registers corresponding to the bits that are set:

- ETH_DMACSR: Channel Status (see Channel status register (ETH_DMACSR))
- ETH_MTLISR: Interrupt Status (see Interrupt status Register (ETH_MTLISR))
- ETH_MACISR: Interrupt Status (see Interrupt status register (ETH_MACISR))

57.8.1 DMA interrupts

Interrupt registers description

The ETH_DMACSR: Channel Status register (see Channel status register (ETH_DMACSR)) captures all the interrupt events of that TxDMA and RxDMA channel. The ETH_DMACIER: Channel Interrupt Enable register (see Channel interrupt enable register (ETH_DMACIER)) contains the corresponding enable bits for each of the interrupt event.

There are two groups of interrupts in the DMA channel namely Normal and Abnormal interrupts. They are indicated by Bits[15:14] of ETH_DMACSR register respectively. The normal group is for events that happen during the normal transfer of packets (TI: transmit interrupt, RI: receive interrupt, TBU: Transmit buffer unavailable) while the abnormal interrupt events are for error events.

Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated. An interrupt is generated only once for multiple events. The driver must scan the Interrupt status register (ETH_DMAISR) for the cause of the interrupt and clear the source in the respective Status register. The interrupt is cleared only when all the bits of Interrupt status register (ETH_DMAISR) are cleared.

Periodic scheduling of Transmit and Receive Interrupt

It is not preferable to generate interrupts for every packet transferred by DMA (RI and TI) for system throughput performance reasons. The Ethernet peripheral gives the flexibility to schedule the interrupt at regular intervals using two methods:

1. Set Interrupt on Completion bit in Transmit descriptor (TDES2[31] in Table 655: TDES2 normal descriptor (read format)) once for every “required” number of packets to be transmitted.
2. Similarly, set the IOC (RDES3[30] in Table 668: RDES3 normal descriptor (read format)) bit only at some specific intervals of Receive descriptors. This way, whenever a received packet transfer to system memory is complete and any of the descriptors used for that packet transfer has the IOC bit set, only then the RI event is generated.
In addition to above, an interrupt timer (ETH_DMACRXIWTR: Channel Rx Interrupt Watchdog Timer) is given for flexible control and periodic scheduling of Receive Interrupt. When this interrupt timer is programmed with a nonzero value, it gets activated as soon as the Rx DMA completes a transfer of a received packet to system memory without asserting the Receive Interrupt because the corresponding interrupt of completion IOC bit (RDES3[30] in Table 668: RDES3 normal descriptor (read format)) is not set. When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding RIE is enabled in ETH_DMACIER register (see Channel interrupt enable register (ETH_DMACIER)). The timer is stopped and cleared before it expires, if the RI is set for a packet transfer whose descriptor’s IOC was set. The timer is reactivated automatically after the next packet transfer is complete without the RI event being generated.

Channel transfer complete interrupt

The Transmit Transfer complete interrupt (TI) and Receive Transfer complete interrupt (RI) is reflected in the Channel Status register (Channel status register (ETH_DMACSR)). The TI bit is set whenever the Tx DMA channel closes the descriptor in which the IOC bit is set (Interrupt On Completion - TDES2[31]). Similarly, the RI bit is set whenever the Rx DMA channel closes the descriptor with the LD bit set and, in any of the descriptors used for transferring that packet, IOC bit is set (Interrupt Enable on completion - RDES3[30]).

The interrupt signal is asserted for the Transfer complete interrupts only when the corresponding interrupts are enabled in the channel interrupt enable register (Channel interrupt enable register (ETH_DMACIER)).

The behavior of the RI/TI interrupts changes depending on the settings of INTM field (bits[17:16]) in the ETH_DMAMR register (DMA mode register (ETH_DMAMR)). Table 652 explains the behavior of the Transfer Complete interrupt.

<table>
<thead>
<tr>
<th>Interrupt Mode</th>
<th>Behavior of TI/RI and interrupt signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTM=0</td>
<td>The TI/RI status signals are set whenever the Transfer complete event is detected. These bits are cleared whenever the software driver writes 1 to these bits. The interrupt signal is asserted whenever the corresponding interrupts are also enabled in ETH_DMACIER register.</td>
</tr>
<tr>
<td>INTM=1</td>
<td>The TI/RI is set as explained above. However, the interrupt is not asserted for any RI/TI event.</td>
</tr>
<tr>
<td>INTM=2</td>
<td>The RI/TI status bits are set whenever the Transfer Complete event is detected and are reset whenever software driver clears them by writing 1. However, if another Transfer complete event is detected before it is cleared (serviced) by the software, then these status bits are automatically set again. However, the interrupt is not generated based on TI/RI.</td>
</tr>
</tbody>
</table>

57.8.2 MTL interrupts

MTL interrupt events are combined with the events in the DMA to generate the interrupt signal.

*Interrupt status Register (ETH_MTLISR)* reports the queue number responsible for the event. ETH_MTLQICSR: Queue Interrupt Control Status must be read for event description.
The MTL interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the Interrupt status Register (ETH_MTLISR) register.

MTL interrupt signal is driven by one of these events:
- Receive Queue Overflow Interrupt
- Transmit Queue Underflow

### 57.8.3 MAC Interrupts

MAC interrupt events are combined with the events in the DMA to generate the interrupt signal.

The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The Interrupt status register (ETH_MACISR) describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the Interrupt status register (ETH_MACISR).

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

MAC interrupt signal is driven by one of these events:
- Receive Status Interrupt
- Transmit Status Interrupt
- Timestamp Interrupt Status
- MMC Interrupt Status
  - MMC Receive Checksum Offload Interrupt Status
  - MMC Transmit Interrupt Status
  - MMC Receive Interrupt Status
- LPI Interrupt Status
- PMT Interrupt Status
- PHY Interrupt

*Note:* Two sidebands signals are generated together with LPI and PMT interrupts: lpi_intr_o and pmt_intr_o. They are used for wake-up event detection at EXTI level.
57.9 Ethernet programming model

This chapter provides the instructions for initializing the DMA or MAC registers in the proper sequence. It contains the following sections:

- DMA initialization (see Section 57.9.1)
- MTL initialization (see Section 57.9.2)
- MAC initialization (see Section 57.9.3)
- Performing Normal Receive and Transmit Operation (see Section 57.9.4)
- Stopping and Starting Transmission (see Section 57.9.5)
- Programming Guidelines for MII Link State Transitions (see Section 57.9.8)
- Programming Guidelines for IEEE 1588 Timestamping (see Section 57.9.9)
- Programming Guidelines for Energy Efficient Ethernet (see Section 57.9.11)
- Programming Guidelines for flexible pulse-per-second (PPS) output (see Section 57.9.12)
- Programming Guidelines for TSO (see Section 57.9.13)
- Programming Guidelines for VLAN filtering on Receive (see Section 57.9.14)

57.9.1 DMA initialization

Complete the following steps to initialize the DMA:

1. Provide a software reset to reset all MAC internal registers and logic (bit 0 of DMA mode register (ETH_DMAMR)).
2. Wait for the completion of the reset process (poll bit 0 of the DMA mode register (ETH_DMAMR), which is cleared when the reset operation is completed).
3. Program the following fields to initialize the System bus mode register (ETH_DMASBMR):
   a) AAL
   b) Fixed burst or undefined burst
   c) Burst mode values in case of AHB bus interface.
4. Create a transmit and a receive descriptor list. In addition, ensure that the receive descriptors are owned by the DMA (set bit 31 of TDES3/RDES3 descriptor). For more information on descriptors, refer to Section 57.10: Descriptors.

Note: Descriptor address from start to end of the ring should not cross the 4GB boundary.

5. Program ETH_DMACTXRLR and ETH_DMACRXRLR registers (see Channel Tx descriptor ring length register (ETH_DMACTXRLR) and Channel Rx descriptor ring length register (ETH_DMACRXRLR)). The programmed ring length must be at least 4.

6. Initialize receive and transmit descriptor list address with the base address of transmit and receive descriptor (Channel Tx descriptor list address register (ETH_DMACTXDLAR), Channel Rx descriptor list address register (ETH_DMACRXDLAR)). In addition, program the transmit and receive tail pointer registers that inform the DMA about the available descriptors (see Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR) and Channel Rx descriptor tail pointer register (ETH_DMACRXDTPR)).

7. Program ETH_DMACCR, ETH_DMACTXCR and ETH_DMACRXCR registers (see Channel control register (ETH_DMACCR), Channel transmit control register (ETH_DMACTXCR) and Channel receive control register (ETH_DMACRXCR)) to
configure the parameters such as the maximum burst-length (PBL) initiated by the DMA, descriptor skip lengths, OSP for TxDMA, RBSZ[13:0] for RxDMA, and so on.

8. Enable the interrupts by programming the ETH_DMACIER register (see Channel interrupt enable register (ETH_DMACIER)).

9. Start the Receive and Transmit DMAs by setting SR (bit 0) of Channel receive control register (ETH_DMAMCRXCR) and ST (bit 0) of the ETH_DMAMCTR (see Channel transmit control register (ETH_DMAMCTXCR)).

57.9.2 MTL initialization

Complete the following steps to initialize the MTL registers:

1. Program the following fields to initialize the operating mode in Tx queue operating mode Register (ETH_MTLTXQOMR).
   a) Transmit Store And Forward (TSF) or Transmit Threshold Control (TTC) if the Threshold mode is used.
   b) Transmit Queue Enable (TXQEN) to value 2'b10 to enable Transmit Queue 0.
   c) Transmit Queue Size (TQS).

2. Program the following fields to initialize the operating mode in the ETH_MTLRXQOMR register (see Rx queue operating mode register (ETH_MTLRXQOMR)):
   a) Receive Store and Forward (RSF) or RTC if Threshold mode is used.
   b) Flow Control Activation and De-activation thresholds for MTL Receive FIFO (RFA and RFD).
   c) Error Packet and undersized good Packet forwarding enable (FEP and FUP).
   d) Receive Queue Size (RQS).

57.9.3 MAC initialization

The MAC configuration registers establish the operating mode of the MAC. If possible, these registers must be initialized before initializing the DMA. The following MAC Initialization operations can also be performed after DMA initialization. If the MAC initialization is complete before the DMA is configured, enable the MAC receiver (last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the Rx FIFO and overflow.

1. Provide the MAC address registers: MAC Address x low register (ETH_MACAxLR), MAC Address 0 high register (ETH_MACA0HR) and MAC Address x high register (ETH_MACAxHR).

2. Program the following fields to set the appropriate filters for the incoming frames in the Packet filtering control register (ETH_MACPFR):
   a) Receive All.
   b) Promiscuous mode.
   c) Hash or Perfect Filter.
   d) Unicast, multicast, broadcast, and control frames filter settings.

3. Program the following fields for proper flow control in the Tx Queue flow control register (ETH_MACQTXFCR):
   a) Pause time and other Pause frame control bits.
   b) Transmit Flow control bits.
   c) Flow Control Busy.
4. Program the Interrupt enable register (ETH_MACIER) as required, if it is applicable for your configuration.

5. Program the appropriate fields in the Operating mode configuration register (ETH_MACCR) register. For example, Inter-packet gap while transmission and jabber disable.

6. Set bit 0 and 1 in Operating mode configuration register (ETH_MACCR) register to start the MAC transmitter and receiver.

To support Jumbo Transmit/Receive packets, follow these steps:
   • In the Operating mode configuration register (ETH_MACCR)
     a) Set JE bit to 1.
     b) Set JD and WD bits to 0 to avoid giant packet error reporting.
     c) Set GPSLCE bit to 1
     d) Set GPSL bitfield of the Extended operating mode configuration register (ETH_MACECR) to a value > 9026

To support Transmit/Receive packets, up to 16K, follow these steps:
   • In the Operating mode configuration register (ETH_MACCR)
     a) Set JD and WD bits to 1 to avoid giant packet error reporting.
     b) Set GPSLCE bit to 1.
     c) Set GPSL bitfield of the Extended operating mode configuration register (ETH_MACECR) to 16383.

57.9.4 Performing normal receive and transmit operation

For normal operation, complete the following steps:
1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptor by reading the status of the descriptor owned by the Host (either transmit or receive).

2. Set the descriptors to appropriate values. Make sure that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.

3. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into Suspend state. The transmission or reception can be resumed by freeing the descriptors and writing the ETH_DMACTXDTPR (see Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR)) and ETH_DMACRXDTPR (see Channel Rx descriptor tail pointer register (ETH_DMACRXDTPR)).

4. In debug mode, the values of the current host transmitter or receiver descriptor address pointer can be read in ETH_DMACCATXDR and ETH_DMACCARXDR registers (see Channel current application transmit descriptor register (ETH_DMACCATXDR) and Channel current application receive descriptor register (ETH_DMACCARXDR)).

5. In debug mode, the values of the current host transmit buffer address pointer and receive buffer address pointer can be read in ETH_DMACCATXDR and ETH_DMACCARXDR registers (see Channel current application transmit descriptor register (ETH_DMACCATXDR) and Channel current application receive descriptor register (ETH_DMACCARXDR)).
57.9.5 Stopping and starting transmission

Complete the following steps to pause the transmission for some time:

1. Disable the Transmit DMA (if applicable) by clearing Bit 0 (ST) of ETH_DMACTXCR register (see Channel transmit control register (ETH_DMACTXCR)).
2. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of Tx queue debug register (ETH_MTLTXQDR) (TRCSTS[1:0] is not 01 and TXQSTS = 0).
3. Disable the MAC transmitter and MAC receiver by clearing RE and TE bits of the Operating mode configuration register (ETH_MACCR) Register.
4. Disable the Receive DMA (if applicable), after making sure that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of Tx queue debug register (ETH_MTLTXQDR), PRXQ=0 and RXQSTS[1:0] = 00).
5. Make sure that both Tx queue and Rx queue are empty (TXQSTS is 0 in Tx queue debug register (ETH_MTLTXQDR) and RXQSTS[1:0] is set to 00).
6. To restart the operation, first start the DMAs, and then enable the MAC Transmitter and Receiver.

**Note:** Do not change the configuration (such as duplex mode, speed, port, or loopback) when the MAC is actively transmitting or receiving. These parameters are changed by software only when the MAC transmitter and receiver are not active. Similarly, do not change the DMA-related configuration when Transmit and Receive DMA are active.

57.9.6 Programming guidelines for switching to new descriptor list in RxDMA

Switching to a new descriptor list is different in the Rx DMA compared to the Tx DMA. Switching to a new descriptor list is permitted when the RxDMA is in Suspend state, as explained below:

- Generally, RxDMA prepares the descriptors in advance.
- If the RxDMA goes to Suspend state due to descriptors not being available, a major failure occurs (software is not able to free the filled-up descriptors/buffers). If this issue is not rectified immediately, frames are lost because of an Rx FIFO overflow. Therefore, the software is allowed to create a new descriptor list and program the RxDMA to start using it immediately, without going into Stop state.

57.9.7 Programming guidelines for switching the AHB clock frequency

To dynamically change the AHB clock frequency (without applying soft reset or hard reset), follow these steps:

1. Disable the Transmit DMA (if applicable) and wait for any previous frame transmissions to complete. When the frame transmissions is complete, the Tx FIFO becomes empty and the Tx DMA enters Stop state. The Tx FIFO status is given in the Tx queue debug
register (ETH_MTLTXQDR) and the status of DMA is given in Debug status register (ETH_DMADSR).

2. Disable the MAC transmitter and the MAC receiver by clearing the appropriate bits in Operating mode configuration register (ETH_MACCR).

3. Disable the Receive DMA (if applicable) after making sure that the data in the Rx FIFO is transferred to the system memory. The Rx FIFO empty status is given in Rx queue debug register (ETH_MTLRXQDR).

4. Ensure that the application does not perform any register read or write operation.

5. Change the frequency of the AHB clock.

6. Enable the MAC Transmitter or the MAC Receiver and the Transmit or Receive DMA. These steps ensure that no valid data is present in the Tx FIFO or Rx FIFO at the time of clock frequency switching and prevent any data corruption.

57.9.8 Programming guidelines for MII link state transitions

Transmit and Receive clocks are running when the link is down

Complete the following steps when the link is down while the Transmit and Receive clocks are running:

1. Disable the Transmit DMA (if applicable) by clearing bit 0 (ST) of Channel control register (ETH_DMACCR).

2. Disable the MAC receiver by clearing RE bit of Operating mode configuration register (ETH_MACCR).

3. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of Tx queue debug register (ETH_MTLTXQDR) (TRCSTS[1:0] is not 01).

   or

   Flush the Tx FIFO for faster empty operation.

4. Disable the MAC transmitter by clearing TE bit of the Operating mode configuration register (ETH_MACCR) Register.

5. Make sure that both Tx and Rx queues are empty (TXQSTS is set to 0 in Tx queue debug register (ETH_MTLTXQDR) and RXQSTS[1:0] to 00 in Rx queue debug register (ETH_MTLRXQDR)).

6. After the link is up, read the PHY registers to identify the latest configuration and program the MAC registers accordingly.

7. Restart the operation by starting the Tx DMA. Then enable the MAC Transmitter and Receiver.

   The Rx DMA does not need to be enabled: since the Receiver is disabled, there are no data in the Rx FIFO.
Transmit and Receive clocks are stopped when the link is down

Complete the following steps when the link is down and the Transmit and Receive clocks are stopped:

1. Disable the MAC Transmitter and Receiver by clearing RE and TE bits in the Operating mode configuration register (ETH_MACCR). This does not take immediate effect as the clocks are absent.
2. Wait till the link is up and the clocks are restored.
3. Wait until the transfer of any partial frame is complete if any was ongoing when the Transmit/Receive clock is stopped. This can be checked by reading the Debug register (ETH_MACDR) (all bits should be set to 0). Some old packets may still remain in the TXFIFO as the MAC Transmitter is stopped.
4. Read the PHY registers to identify the latest operating mode and program the MAC registers accordingly.
5. Restart the MAC Transmitter and Receiver by setting RE and TE bits.

57.9.9 Programming guidelines for IEEE 1588 timestamping

Initializing the System time generation

The timestamp feature can be enabled by setting bit 0 of the Timestamp control Register (ETH_MACTSCR). However, it is essential that the timestamp counter is initialized after this bit is set. Complete the following steps to perform the peripheral initialization:

1. Mask the Timestamp Trigger interrupt by clearing bit 12 of Interrupt enable register (ETH_MACIER).
2. Set bit 0 of Timestamp control Register (ETH_MACTSCR) to enable timestamping.
3. Program Subsecond increment register (ETH_MACSSIR) based on the PTP clock frequency.
4. If you use the Fine Correction method, program Timestamp addend register (ETH_MACTSAR) and set bit 5 of Timestamp control Register (ETH_MACTSCR).
5. Poll the Timestamp control Register (ETH_MACTSCR) until bit 5 is cleared.
6. Program bit 1 of Timestamp control Register (ETH_MACTSCR) to select the Fine Update method (if required).
7. Program System time seconds update register (ETH_MACSTSUR) and System time nanoseconds update register (ETH_MACSTNUR) with the appropriate time value.
8. Set bit 2 in Timestamp control Register (ETH_MACTSCR). The timestamp counter starts as soon as it is initialized with the value written in the timestamp update registers. If one-step timestamping is required:
   a) Enable one-step timestamping by programming bit 27 of the TDES3 Context Descriptor.
   b) Program Timestamp Ingress asymmetric correction register (ETH_MACTSIACR) to update the correction field in PDelay_Req PTP messages.
9. Enable the MAC receiver and transmitter for proper timestamping.

Note: If timestamp operation is disabled by clearing bit 0 of Timestamp control Register (ETH_MACTSCR), repeat all these steps to restart the timestamp operation.
System time correction

To synchronize or update the system time in one shot (coarse correction method), complete the following steps:

1. Set the offset (positive or negative) in the timestamp update registers (System time seconds update register (ETH_MACSTSUR) and System time nanoseconds update register (ETH_MACSTNUR)).
2. Set bit 3 (TSUPDT) of the Timestamp control Register (ETH_MACTSCR).

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

1. With the help of the algorithm described in Section: System time register module, calculate at which rate you intend to increment or decrement the system time.
2. Update the Timestamp addend register (ETH_MACTSAR) with the new value and set bit 5 of the Timestamp control Register (ETH_MACTSCR) Register.
3. Wait for the time during which you want the new value of the Addend register to be active. This can be done by enabling the Timestamp Trigger interrupt after the system time reaches the target value.
4. Program the required target time in PPS target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTNTR).
5. Enable the Timestamp interrupt in bit 12 of Interrupt enable register (ETH_MACIER).
6. Set bit 4 in Register Timestamp control Register (ETH_MACTSCR).
7. When this trigger generates an interrupt, read Interrupt status register (ETH_MACISR).
8. Reprogram Timestamp addend register (ETH_MACTSAR) with the old value and set bit 5 again.

57.9.10 Programming guidelines for PTP offload feature

Programming guidelines to enable automatic periodic generation of PTP sync messages

Follow these steps to enable automatic periodic generation of PTP sync messages:

1. Program SNAPYPSE, TSMSTRENA, and TSEVNTEN values of Timestamp control Register (ETH_MACTSCR) to 0, 1, and 1 respectively, to configure the node as Ordinary or Boundary Master (1, 1, and 1 for Transparent Master).
2. Program the PTOEN bit and DN field of PTP Offload control register (ETH_MACPOCR) to enable PTP Offload feature and domain number to send in egress PTP Sync message.
3. Program the ASYNCE bit of PTP Offload control register (ETH_MACPOCR) to enable periodic generation of PTP Sync messages.
4. Program the 80-bit Source Port Identity in PTP Source Port Identity 0 Register (ETH_MACSPI0R), PTP Source port identity 1 register (ETH_MACSPI1R) and PTP
**Source port identity 2 register (ETH_MACSPI2R)** to send in egress PTP Sync message.

5. Program the LSI field of **Log message interval register (ETH_MACLMIR)** to program the periodicity of the PTP Sync messages.

For example, a value of 1 corresponds to $2^1$ which translates to PTP Sync message every 2 seconds, and a value of 0xFF (twos complement of -1) corresponds to $2^{-1}$ which translates to PTP Sync message every 0.536 seconds.

6. Program the TSIE bit of **Interrupt enable register (ETH_MACIER)** to enable generation of Timestamp interrupt.

7. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in **Timestamp status register (ETH_MACTSSR)**. It indicates that the timestamp for PTP Sync message is captured in **Tx timestamp status seconds register (ETH_MACTXTSSSR)** and **Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)**.

**Programming guidelines to enable periodic generation of PTP Pdelay_Req messages**

Follow these steps to enable automatic periodic generation of PTP Pdelay_Req messages

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of **Timestamp control Register (ETH_MACTSCR)** to 1, 0, and 1 respectively to configure the node as Transparent Slave (1, 1, and 1 for Transparent Master OR 3, X, and X for Peer-to-Peer Transparent).

2. Program the PTOEN bit and DN field of **PTP Offload control register (ETH_MACPOCR)** to enable PTP Offload feature and domain number to send in egress PTP Pdelay_Req message.

3. Program the APDREQEN bit of **PTP Offload control register (ETH_MACPOCR)** to enable periodic generation of PTP Pdelay_Req messages.

4. Program the 80-bit Source Port Identity in **PTP Source Port Identity 0 Register (ETH_MACSPI0R)**, **PTP Source port identity 1 register (ETH_MACSPI1R)** and **PTP Source port identity 2 register (ETH_MACSPI2R)** to send in egress PTP Pdelay_Req message.

5. Program the LMPDRI field of **Log message interval register (ETH_MACLMIR)** to program the periodicity of the PTP Pdelay_Req messages.

For example, a value of 1 corresponds to $2^1$ which translates to PTP Pdelay_Req message every 2 seconds, and a value of 0xFF (twos complement of -1) corresponds to $2^{-1}$ which translates to PTP Pdelay_Req message every 0.536 seconds.

6. Program the TSIE bit of **Interrupt enable register (ETH_MACIER)** to enable generation of Timestamp interrupt.

7. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in **Timestamp status register (ETH_MACTSSR)**. It indicates that the timestamp for PTP Sync message is captured in **Tx timestamp status seconds register (ETH_MACTXTSSSR)** and **Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)**.

**Programming guidelines to enable the generation of PTP response messages for Ordinary or Boundary Master mode**

Follow these steps to enable the generation of PTP response messages for Ordinary or Boundary Master mode (Periodic PTP Sync messages generated and PTP Delay_Resp message generated in response to PTP Delay_Req message):

- **Source port identity 2 register (ETH_MACSPI2R)** to send in egress PTP Sync message.
1. Program SNAPTYPESEL, TSMSTRENA and TSEVNTENA fields of Timestamp control Register (ETH_MACTSCR) to 0, 1, and 1 respectively.
2. Program the PTOEN bit and DN field of PTP Offload control register (ETH_MACPOCR) to enable PTP Offload feature and domain number to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
3. Program the 80-bit Source Port Identity in PTP Source Port Identity 0 Register (ETH_MACSPI0R), PTP Source port identity 1 register (ETH_MACSPI1R) and PTP Source port identity 2 register (ETH_MACSPI2R) to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
4. Program the DRSYNCR and LSI fields in Log message interval register (ETH_MACLIMR). The sum of both fields is updated in logMinDelayReqInterval field of PTP Delay_Resp message.

Programming guidelines to enable the generation of PTP response messages for Ordinary or Boundary Slave mode

Follow these steps to enable generation of PTP response messages for Ordinary or Boundary Slave mode (PTP Delay_Req message generated in response to PTP Sync message):

1. Program SNAPTYPESEL, TSMSTRENA and TSEVNTENA fields of Timestamp control Register (ETH_MACTSCR) to 0, 0, and 1 respectively.
2. Program the PTOEN bit and DN field of PTP Offload control register (ETH_MACPOCR) to enable PTP Offload feature and domain Number to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
3. Program the 80-bit Source Port Identity in PTP Source Port Identity 0 Register (ETH_MACSPI0R), PTP Source port identity 1 register (ETH_MACSPI1R) and PTP Source port identity 2 register (ETH_MACSPI2R) to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
4. Program the DRSYNCR field in Log message interval register (ETH_MACLIMR) to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages.

Programming guidelines to enable the generation of PTP response messages for Transparent Slave mode

Follow these steps to enable generation of PTP response messages for Transparent Slave mode (PTP Delay_Req message generated in response to PTP Sync message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated)

1. Program SNAPTYPESEL, TSMSTRENA and TSEVNTENA fields of Timestamp control Register (ETH_MACTSCR) to 1, 0, and 1 respectively.
2. Program the PTOEN bit and DN field of PTP Offload control register (ETH_MACPOCR) to enable PTP Offload feature and domain Number to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.
3. Program the 80-bit Source Port Identity in PTP Source Port Identity 0 Register (ETH_MACSPI0R), PTP Source port identity 1 register (ETH_MACSPI1R) and PTP Source port identity 2 register (ETH_MACSPI2R) to match with ingress PTP Sync or
Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.

4. Program the DRSYNCR and LMPDRI fields in Log message interval register (ETH_MACLMIR) to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages and periodicity of the PTP Pdelay_Req messages.

5. Program the TSIE bit of Interrupt enable register (ETH_MACIER) to enable generation of Timestamp interrupt.

6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in Timestamp status register (ETH_MACTSSSR). It indicates that the timestamp for PTP Sync message is captured in Tx timestamp status seconds register (ETH_MACTXTSSSR) and Tx timestamp status nanoseconds register (ETH_MACTXTSSNR) for egress PTP Pdelay_Req and Pdelay_Resp messages.

Programming guidelines to enable the generation of PTP response messages for Transparent Master mode

Follow these steps to enable generation of PTP response messages for Transparent Master mode (PTP Delay_Resp message generated in response to PTP Delay_Req message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req or Sync messages generated):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of Timestamp control Register (ETH_MACTSCR) to 1, 1, and 1 respectively.

2. Program the PTOEN bit and DN field of PTP Offload control register (ETH_MACPOCR) to enable PTP Offload feature and domain number to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.

3. Program the 80-bit Source Port Identity in PTP Source Port Identity 0 Register (ETH_MACSPI0R), PTP Source port identity 1 register (ETH_MACSPI1R) and PTP Source port identity 2 register (ETH_MACSPI2R) to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.

4. Program the DRSYNCR, LSI and LMPDRI fields in Log message interval register (ETH_MACLMIR), the sum of DRSYNCR and LSI is updated in logMinDelayReqInterval field of PTP Delay_Resp message and periodicity of the PTP Sync or Pdelay_Req messages.

5. Program the TSIE bit of Interrupt enable register (ETH_MACIER) to enable generation of Timestamp interrupt.

6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in Timestamp status register (ETH_MACTSSSR). It indicates that the timestamp for PTP Sync message is captured in Tx timestamp status seconds register (ETH_MACTXTSSSR) and Tx timestamp status nanoseconds register (ETH_MACTXTSSNR) for egress PTP Sync, Pdelay_Req and Pdelay_Resp messages.

Programming guidelines to enable the generation of PTP response messages for Peer-to-Peer Transparent mode

Follow these steps to enable generation of PTP response messages for Peer-to-Peer Transparent mode (PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated):
1. Program the SNAPTYPSEL, TSMSTREN and TSEVNTENA fields of Timestamp control Register (ETH_MACTSCR) to 3, X, and X respectively.

2. Program the PTOEN bit and DN field of PTP Offload control register (ETH_MACPOCR) to enable PTP Offload feature and domain Number to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.

3. Program the 80-bit Source Port Identity in PTP Source Port Identity 0 Register (ETH_MACSPI0R), PTP Source port identity 1 register (ETH_MACSPI1R) and PTP Source port identity 2 register (ETH_MACSPI2R) to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.

4. Program the LMPDRI field in Log message interval register (ETH_MACLMIR) to indicate periodicity of the PTP Pdelay_Req messages

5. Program the TSIE bit of Interrupt enable register (ETH_MACIER) to enable generation of Timestamp interrupt

6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in Timestamp status register (ETH_MACTSSSR). It indicates that the timestamp for PTP Sync message is captured in Tx timestamp status seconds register (ETH_MACTXTSSSR) and Tx timestamp status nanoseconds register (ETH_MACTXTSNSR) for egress PTP Pdelay_Req and Pdelay_Resp messages.

57.9.11 Programming guidelines for Energy Efficient Ethernet (EEE)

Entering and exiting Tx LPI mode

EEE enables the IEEE 802.3 Media Access Control (MAC) sublayer along with a family of physical layers to operate in the Low-power idle (LPI) mode. In the Transmit path, the software must set the LPIEN bit of the LPI control and status register (ETH_MACLCSR) to indicate to the MAC to stop transmission and initiate the LPI protocol.

Complete the following steps during MAC initialization:

1. Read the PHY register through the MDIO interface and check if the remote end has the EEE capability. Then negotiate the timer values.

2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop Rx clock in LPI mode or not).

3. Program bits 25 to 16 and bits 15 to 0 in LPI timers control register (ETH_MACLTCR).

4. Read the PHY link status by using the MDIO interface and update bit 17 of LPI control and status register (ETH_MACLCSR).

Update LPI control and status register (ETH_MACLCSR) accordingly. This update should be done whenever the link status in the PHY chip changes.

5. Program One-microsecond-tick counter register (ETH_MAC1USTCR) as per the frequency of the clock used for accessing the CSR slave port.

6. Program the LPIET bit in the LPI entry timer register (ETH_MACLETR) with the IDLE time for which the MAC should wait before entering the LPI state on its own.
7. Set LPITE and LPITXA (bits 20 to 19) of LPI control and status register (ETH_MACLCSR) to enable LPI auto-entry and MAC auto-exit from LPI state.

8. Set bit 16 of LPI control and status register (ETH_MACLCSR) to put the MAC transmitter in LPI state.
   The MAC enters the LPI state when all scheduled packets are completed. It remains IDLE for the time indicated by LPIET bits. It sets the TLPIEN (bit 0) after entering LPI state.

9. When a packet transmission is scheduled (when the TxDMA exits IDLE state or when a packet is presented at ATI or MTI interface), the MAC Transmitter automatically exits LPI state. It waits for TWT time before setting the TLPIEX interrupt status bit and then resume the packet transmission.

10. The MAC Transmitter enters again LPI state if it remains IDLE for LPIET time. It then sets the TLPIEN bit and the entry-exit cycle continues.

11. Reset LPNIE bit if the application needs to override the auto-entry/exit modes and directly exit the MAC Transmitter from LPI state.

**Note:** To make sure the MAC enters the LPI state only after the transmission of all the queued frames in the Tx FIFO is complete, set LPITXA bit in LPI control and status register (ETH_MACLCSR).

To switch off the CSR clock or power to the rest of the system during the LPI state, wait for the TLPIEN interrupt of LPI control and status register (ETH_MACLCSR) to be generated. Restore the clocks before performing step 6 when you want to come out of the LPI state.

### Gating Off the CSR Clock in the LPI mode

You can gate off the CSR clock to save the power when the MAC is in the Low-Power Idle (LPI) mode.

#### Gating off the CSR clock in the Rx LPI mode

The following operations are performed when the MAC receives the LPI pattern from the PHY:

1. The MAC RX enters the LPI mode and the Rx LPI entry interrupt status (RLPIEN interrupt of LPI control and status register (ETH_MACLCSR)) is set.
2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the LPI control and status register (ETH_MACLCSR).

After the sbd_intr_o interrupt is asserted and the MAC Tx is also in the LPI mode, the CSR clock can be gated off. If the MAC TX is not in LPI mode when the CSR clock is gated off, the events on the MAC transmitter do not get reported or updated in the CSR. To restore the CSR clock, wait for the LPI exit indication from the PHY after which the MAC asserts the LPI exit interrupt on lpi_intr_o (synchronous to clk_rx_i). The lpi_intr_o interrupt is cleared when LPI control and status register (ETH_MACLCSR) is read.

#### Gating off the CSR clock in the Tx LPI mode

The following operations are performed when bit 16 (LPIEN) of LPI control and status register (ETH_MACLCSR) is set:

1. The Transmit LPI Entry interrupt (TLPIEN bit of LPI control and status register (ETH_MACLCSR)) is set.
2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the LPI control and status register (ETH_MACLCSR).
After the sbd_intr_o interrupt is asserted and the MAC RX is also in the LPI mode, the CSR
clock can be gated off. If the MAC RX is not in LPI mode when the CSR clock is gated off,
the events on the MAC receiver do not get reported or updated in the CSR. To restore the
CSR clock, switch on the CSR clock when the MAC has exited TX LPI mode. After the CSR
clock is resumed, reset bit 16 (LPIEN) of LPI control and status register (ETH_MACLCSR)
to exit the MAC from LPI mode.

57.9.12 Programming guidelines for flexible pulse-per-second (PPS) output

Generating a single pulse on PPS

To generate a single pulse on PPS:
1. Program TRGTMODSEL[1:0] bit to 11 or 10 (for interrupt) in PPS control register
(ETH_MACPPSCR). This instructs the MAC to use the Target Time registers (PPS
target time seconds register (ETH_MACPPTTSR) and PPS target time nanoseconds
register (ETH_MACPPTTNR)) as start time of PPS signal output.
2. Program the start time value in the Target Time registers (register PPS target time
seconds register (ETH_MACPPTTSR) and PPS target time nanoseconds register
(ETH_MACPPTTNR)).
3. Program the width of the PPS signal output in PPS width register (ETH_MACPPSWR)
Register.
4. Program PPSCMD[3:0] of PPS control register (ETH_MACPPSCR) to 0001. This
instructs the MAC to generate a single pulse on the PPS signal output at the time
programmed in the Target Time registers.

Generating next pulse on PPS

When the PPSCMD is executed (PPSCMD bits = 0), you can cancel the pulse generation by
giving the Cancel Start Command (PPSCMD=0011) before the programmed start time has
elapsed. You can also program the behavior of the next pulse in advance. To program the
next pulse:
1. Program the start time for the next pulse in the Target Time registers. This time should
be higher than the time at which the falling edge occurs for the previous pulse.
2. Program the width of the next PPS signal output in PPS width register
(ETH_MACPPSWR).
3. Program PPSCMD[3:0] bits of PPS control register (ETH_MACPPSCR) to generate a
single pulse after the previous pulse is deasserted. This instructs the MAC to generate
a single pulse on the PPS signal output at the time programmed in Target Time
registers.

If this command is given before the previous pulse becomes low, then the new
command overwrites the previous command and the peripheral may generate only 1
extended pulse.

Generating a pulse train on PPS

To generate a pulse train on PPS:
1. Program TRGTMODSEL[1:0] bits to 11 or 10 (for interrupt) in PPS control register
(ETH_MACPPSCR). This instructs the MAC to use the Target Time registers (PPS
2. Program the start time value in the Target Time registers (register PPS target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTTNR)).

3. Program the interval value between the train of pulses on the PPS signal output in PPS interval register (ETH_MACPPSIR).

4. Program the stop value in the Target Time registers. Ensure that TSTRBUSY bit in PPS target time nanoseconds register (ETH_MACPPSTTNR) is reset before programming the Target Time registers again.

5. Program PPSCMD[3:0] bits in PPS control register (ETH_MACPPSCR) to 0010. This instructs the MAC to generate a train of pulses on the PPS signal output at the start time programmed in Target Time registers.

6. Program the stop value in the Target Time registers. Ensure that TSTRBUSY bit in PPS target time nanoseconds register (ETH_MACPPSTTNR) is reset before programming the Target Time registers again.

7. Program the PPSCMD[3:0] bits in PPS control register (ETH_MACPPSCR) to 0100 to stop the train of pulses on PPS signal output after the programmed stop time specified at step 6 has elapsed.

The pulse train can be stopped at any time by programming 0101 in the PPSCMD[3:0] field. Similarly, the Stop Pulse train command (given in Step 7) can be canceled by programming PPSCMD[3:0] bits to 0110 before the time (programmed at step 6) has elapsed. The pulse train generation can be stopped by programming PPSCMD[3:0] to 0011 before the start time programmed at step 2) has elapsed.

**Generating an interrupt without affecting the PPS**

TRGTMODSEL[1:0] bits in PPS control register (ETH_MACPPSCR) enable you to program the Target Time registers (PPS target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTTNR)) to do any one of the following:

- Generate only interrupts.
- Generate interrupts and the PPS start and stop time.
- Generate only PPS start and stop time.

To program the Target Time registers to generate only interrupt event:

1. Program TRGTMODSEL[1:0] bits of PPS control register (ETH_MACPPSCR) to 00 (for interrupt). This instructs the MAC to use the Target Time registers for target time interrupt.

2. Program a target time value in the Target Time registers. This instructs the MAC to generate an interrupt when the target time elapses.

If TRGTMODSEL[1:0] bits are changed (for example, to control the PPS), then the interrupt generation is overwritten with the new mode and new programmed Target Time register value.

**Note:** The TSTRGTER0 bit in Timestamp status register (ETH_MACTSSR) is set when the programmed target time is smaller (that is corresponds to a time in the past) compared to
the system time in the System time seconds register (ETH_MACSTSR) and System time
nanoseconds register (ETH_MACSTNR).

An interrupt is generated (sbd_intr_o) if the TSIE bit in the Interrupt enable register
(ETH_MACIER) is set.

Therefore, to avoid unwanted interrupt, the correct writing order is as follow:
1. PPS target time nanoseconds register (ETH_MACPPSTTNR).
2. PPS target time seconds register (ETH_MACPPSTTSR).
3. PPS interval register (ETH_MACPPSIR).
4. PPS width register (ETH_MACPPSWR).
5. PPSCTRL[3:0] and PPSEN0 bitfields of PPS control register
   (ETH_MACPPSCR).

57.9.13 Programming guidelines for TSO

The TCP Segmentation Offload (TSO) engine is used to offload the TCP segmentation
functions to the hardware. To program the TSO, set the TSE bit to enable TCP packet
segmentation, and program descriptor fields to enable TSO for the current packet.

Follow the steps below to program TSO:
1. Program TSE bit of the corresponding Channel transmit control register
   (ETH_DMACTXCR) to enable TCP packet segmentation in that DMA.
2. In addition to the normal transfer descriptor setting, the following descriptor fields must
   be programmed to enable TSO for the current packet:
   a) Enable TSE of TDES3 (bit 18).
   b) Program the length of the unsegmented TCP/IP packet payload in bits 17 to 0 of
      TDES3, and the TCP header in bits 22 to 19 of TDES3.
   c) Program the maximum size of the segment in:
      – MSS[13:0] of Channel control register (ETH_DMACCR)
      – or MSS in the context descriptor
         If MSS[13:0] field is programmed in both Channel control register
         (ETH_DMACCR) and in the context descriptor, the latest software programmed
         sequence is considered.
3. The unsegmented TCP/IP packet header should be stored in Buffer 1 of the first
   descriptor. This buffer must not hold any payload bytes. The payload is allocated to
   Buffer 2 and the buffers of the subsequent descriptors.

Caution: If TSE is enabled in TDES3 for a non-TCP-IP packet, the result is unpredictable.
57.9.14 Programming guidelines to perform VLAN filtering on the receive

Follow the sequence below to perform VLAN filtering on the receiver:

1. Program VLAN tag register (ETH_MACVTR) for the following bit to select the filtering method:
   - ETV: Enable 12-bit VLAN Tag Comparison or 16-bit VLAN Tag comparison.
   - VTHM: VLAN Tag Hash Table Match Enable.
   - ERIVLT: Enable inner VLAN Tag or outer VLAN Tag (to enable the inner or outer VLAN Tag filtering, Double VLAN Processing should be enabled by setting EDVLP)
   - ERSVLM: Enable Receive S-VLAN Match or C-VLAN match (for S-VLAN processing to be enabled, set ESVL)
   - DOVLTC: Ignores VLAN Type for Tag Match
   - VTIM: to enable VLAN Tag Inverse Match instead of the normal VLAN Tag matching

2. Program VL bit in VLAN tag register (ETH_MACVTR) for the 12-bit or 16-bit VLAN tag.

3. If VLAN tag Hash filtering is enabled, program VLAN Hash table register (ETH_MACVHTR):
   - When the ETV bit is reset, the upper 4 bits of the calculated CRC-32 of VLAN tag are inverted and used to index the content of the VLAN Hash table register (ETH_MACVHTR).
   - When ETV bit is set, the upper 4 bits of the calculated CRC-32 of VLAN tag are used to index the content of VLAN Hash table register (ETH_MACVHTR).

For example, when ETV bit is set, a hash value of 0b1000 selects bit 8 of the VLAN Hash table. When ETV bit is reset, a hash value of 0b1000 selects bit 7 of the VLAN Hash table.

57.10 Descriptors

57.10.1 Descriptor overview

In the Ethernet peripheral, the DMA transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory (SRAM). The following two types of descriptors are supported:

- **Normal descriptors**
  The normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.

- **Context descriptors**
  The context descriptors are used to provide control information applicable to the packet to be transmitted.

Each normal descriptor contains two buffers and two address pointers. These buffers enable the adapter port to be compatible with various types of memory management schemes.

There is no limit to the number of descriptors that can be used for a single packet.
57.10.2 Descriptor structure

The Ethernet peripheral supports the ring structure for DMA descriptors.

In a ring structure, descriptors are separated by the 32-bit word number programmed in the DSL field of the Channel control register (ETH_DMACCR). The application needs to program the total ring length, that is the total number of descriptors in ring span, in the following registers of a DMA channel:

- Channel Tx descriptor ring length register (ETH_DMACTXRLR)
- Channel Rx descriptor ring length register (ETH_DMACRXRLR)

The Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR) or Channel Rx descriptor tail pointer register (ETH_DMACRXDTPR) contains the pointer to the descriptor address (N). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (N – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

**Current Descriptor Pointer == Descriptor Tail Pointer;**

The DMA enters the Suspend state when this condition occurs. The application must perform a write operation to the Descriptor tail pointer register and update the tail pointer so that the following condition is met:

**Current Descriptor Pointer < Descriptor Tail Pointer;**
The DMA automatically wraps around the base address when the end of ring is reached, as shown in Figure 821: DMA descriptor ring.

Figure 821. DMA descriptor ring

For descriptors owned by the application, the OWN bit of DES3 is reset to 0.

For descriptors owned by the DMA, the OWN bit is set to 1.

At the beginning, if the application has only one descriptor, it sets the last descriptor address (tail pointer) to Descriptor Base Address + 1. The DMA then processes the first descriptor and waits for the application to increment the tail pointer.
57.10.3 Transmit descriptor

The Ethernet peripheral DMA requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor features control fields which can be used to manage the MAC operation on per-transmit packet basis. The Transmit normal descriptor has the following two formats: Read format and Write-back format.

Transmit normal descriptor (read format)

*Figure 822* shows the Read format for Transmit normal descriptor. *Table 653* to *Table 656* provide a detailed description of all Transmit normal descriptors (read format).

### Figure 822. Transmit descriptor (read format)

![Transmit descriptor (read format)](image)

- **TDES0 normal descriptor (read format)**

*Table 653. TDES0 normal descriptor (read format)*

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>BUF1AP</td>
<td>Buffer 1 Address Pointer or TSO Header Address Pointer</td>
</tr>
</tbody>
</table>

- These bits indicate either the physical address of Buffer 1 or the TSO Header Address pointer when the following bits are set:
  - TSE bit of TDES3
  - FD bit of TDES3
• TDES1 normal descriptor (read format)

Table 654. TDES1 normal descriptor (read format)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | BUF2AP | **Buffer 2 or Buffer 1 Address Pointer:**  
These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. There is no limitation to the buffer address alignment. |

• TDES2 normal descriptor (read format)

Table 655. TDES2 normal descriptor (read format)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31   | IOC  | **Interrupt on completion:**  
This bit sets the TI bit in the Channel status register (ETH_DMACSR) when the present packet transmission is complete. |
| 30   | TTSE | **Transmit Timestamp Enable**  
This bit enables the IEEE1588 timestamping for Transmit packet referenced by the descriptor. |
| 29:16| B2L  | **Buffer 2 Length**  
The driver sets this field. When set, this field indicates Buffer 2 length. |
| 15:14| VTIR | **VLAN Tag Insertion or Replacement:**  
These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC Pad Control bits appropriately when VLAN tag insertion, replacement, or deletion is enabled for the packet. The values of these bits are as follows:  
00: Do not add a VLAN tag.  
01: Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets.  
10: Insert a VLAN tag with the tag value programmed in the VLAN inclusion register (ETH_MACVIR) or context descriptor.  
11: Replace the VLAN tag in packets with the tag value programmed in the VLAN inclusion register (ETH_MACVIR) or context descriptor. This option should be used only with the VLAN packets. |
Ethernet (ETH): media access control (MAC) with DMA controller

Table 655. TDES2 normal descriptor (read format) (continued)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 13:0 | HL or B1L | Header length or buffer 1 length  
For Header length, only bits [9:0] are taken into account. Bits 13 to 0 are applicable only to buffer 1 length.  
If the TCP Segmentation Offload feature is enabled through the TSE bit of TDES3, this field is equal to the header length. When the TSE bit is set in TDES3, the header length includes the length (expressed in bytes) from Ethernet Source address till the end of the TCP header. The maximum header length supported for TSO feature is 1023 bytes.  
If the TCP Segmentation Offload feature is not enabled, this field is equal to Buffer 1 length. |

Table 656. TDES3 normal descriptor (read format)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31   | OWN    | Own bit  
1: the DMA owns the descriptor.  
0: the application owns the descriptor.  
The DMA clears this bit after it completes the transfer of data given in the associated buffer(s). |
| 30   | CTXT   | Context Type  
This bit should be set to 0 for normal descriptor. |
| 29   | FD     | First Descriptor  
When this bit is set, it indicates that the buffer contains the first segment of a packet. |
| 28   | LD     | Last Descriptor  
When this bit is set, it indicates that the buffer contains the last segment of the packet. B1L or B2L field should have a non-zero value. |
Table 656. TDES3 normal descriptor (read format) (continued)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 27:26 | CPC   | CRC Pad Control  
This field controls the CRC and Pad Insertion for Tx packet. It is valid only when the first descriptor bit (TDES3[29]) is set. The values of bits[27:26] are the following:  
00: CRC and Pad Insertion  
The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packets whose length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes.  
01: CRC Insertion (Disable Pad Insertion)  
The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the Transmit buffer, that is, the packet being transferred from the Transmit Buffer is of length greater than or equal to 60 bytes.  
10: Disable CRC Insertion  
The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer.  
11: CRC Replacement  
The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer.  
When the TSE bit is set, the MAC ignores this field because the CRC and pad insertion is always done for segmentation. |
| 25:23 | SAIC  | SA Insertion Control  
These bits request the MAC to add or replace the Source Address field in the Ethernet packet with the value given in the MAC Address 0 register. The application must appropriately set the CRC Pad Control bits when SA Insertion Control is enabled for the packet.  
Bit 25 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement.  
The following list describes the values of Bits[24:23]:  
00: Do not include the source address  
01: Include or insert the source address. For reliable transmission, the application must provide frames without source addresses.  
10: Replace the source address. For reliable transmission, the application must provide frames with source addresses.  
11: Reserved  
These bits are valid when the First Segment control bit (TDES3 [29]) is set. |
| 22:19 | THL   | THL: TCP Header Length  
If the TSE bit is set, this field contains the length of the TCP/UDP header. The minimum value of this field must be 5 for TCP header. THL value must be equal to 2 for UDP header. This field is valid only for the first descriptor. |
| 18    | TSE   | TCP Segmentation Enable  
When this bit is set, the DMA performs the TCP/UDP segmentation for a packet. This bit is valid only if the FD bit is set. |
Transmit normal descriptor (write-back format)

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[28]) is set in the descriptor where the DMA writes back the status and timestamp information for the corresponding Transmit packet. 

Figure 823 shows the write-back format for Transmit normal descriptors. Table 657 to Table 660 provide a detailed description of all Transmit Normal descriptors (Write-Back Format).

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>17:16</td>
<td>CIC/TPL</td>
<td>Checksum Insertion Control or TCP Payload Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>These bits control the checksum calculation and insertion. They can take</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00: Checksum insertion disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01: Only IP header checksum calculation and insertion are enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10: IP header checksum and payload checksum calculation and insertion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>are enabled, but pseudo-header checksum is not calculated in hardware.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11: IP header checksum and payload checksum calculation and insertion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>are enabled, and pseudo-header checksum is calculated in hardware.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This field is valid when the TSE bit is reset. When the TSE bit is set, it</td>
</tr>
<tr>
<td></td>
<td></td>
<td>contains the upper bits [17:16] of the TCP Payload length. This allows the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TCP packet length field to be spanned across TDES3[17:0] to provide</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256 Kbyte packet length support.</td>
</tr>
<tr>
<td>15</td>
<td>TPL</td>
<td>Reserved or TCP Payload Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When the TSE bit is reset, this bit is reserved. When the TSE bit is set,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>this is bit 15 of the TCP payload length [17:0].</td>
</tr>
<tr>
<td>14:0</td>
<td>FL/TPL</td>
<td>Reserved or TCP Payload Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When the TSE bit is set, this field is equal to the lower 15 bits of the TCP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>payload length. This length does not include Ethernet header or TCP/UDP/IP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>header length.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When the TSE bit is reset, this bit is reserved.</td>
</tr>
</tbody>
</table>
Figure 823. Transmit descriptor write-back format

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | TTSL | Transmit Packet Timestamp Low  
The DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding Transmit packet. The DMA writes the timestamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field holds the timestamp only if the Last Segment bit (LS) in the descriptor is set and the Timestamp status (TTSS) bit is set. |

1. This format is only applicable to the last descriptor of a packet.

- TDES0 normal descriptor (write-back format)

- TDES1 normal descriptor (write-back format)

Table 658. TDES1 normal descriptor (write-back format)(1)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | TTSH | Transmit Packet Timestamp High  
The DMA updates this field with the most significant 32 bits of the timestamp captured for corresponding Receive packet. The DMA writes the timestamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set. |

1. This format is only applicable to the last descriptor of a packet.

- TDES2 normal descriptor (write-back format)
### Table 659. TDES2 normal descriptor (write-back format)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

1. This format is only applicable to the last descriptor of a packet.

- TDES3 normal descriptor (write-back format)

```plaintext
<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>OWN</td>
<td>Own bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, it indicates that the DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 0.</td>
</tr>
<tr>
<td>30</td>
<td>CTXT</td>
<td>Context Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit should be set to 0 for normal descriptors.</td>
</tr>
<tr>
<td>29</td>
<td>FD</td>
<td>First Descriptor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit indicates that the buffer contains the first segment of a packet.</td>
</tr>
<tr>
<td>28</td>
<td>LD</td>
<td>Last Descriptor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit is set 1 for last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.</td>
</tr>
<tr>
<td>27:18</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>TTSS</td>
<td>Tx Timestamp Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This status bit indicates that a timestamp has been captured for the corresponding transmit packet. When this bit is set, TDES0 and TDES1 have timestamp values that were captured for the Transmit packet. This field is valid only when the Last Segment control bit (TDES3 [28]) in a descriptor is set.</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
```

---

1. This format is only applicable to the last descriptor of a packet.
## Error Summary

This bit indicates the logical OR of the following bits:
- TDES3[0]: IP Header Error
- TDES3[14]: Jabber Timeout
- TDES3[13]: Packet Flush
- TDES3[12]: Payload Checksum Error
- TDES3[11]: Loss of Carrier
- TDES3[10]: No Carrier
- TDES3[9]: Late Collision
- TDES3[8]: Excessive Collision
- TDES3[3]: Excessive Deferral
- TDES3[2]: Underflow Error

## Jabber Timeout

This bit indicates that the MAC transmitter has experienced a jabber timeout. This bit is set only when the JD bit of the Operating mode configuration register (ETH_MACCR) is not set.

## Packet Flushed

This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.

## Payload Checksum Error

This bit indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either caused by insufficient bytes, as indicated by the Payload Length field of the IP Header, or by the MTL starting to forward the packet to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store-and-forward mode. This error can also occur when a Bus error is detected during packet transfer.

## Loss of Carrier

This bit indicates that Loss of Carrier occurred during packet transmission (that is, the ETH_CRS signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the Half-duplex mode.

## No Carrier

This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.

## Late Collision

This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including Preamble in MII mode). This bit is not valid if Underflow Error is set.
Table 660. TDES3 normal descriptor (write-back format)(1) (continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 8   | EC       | Excessive Collision  
This bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the Operating mode configuration register (ETH_MACCR), this bit is set after first collision and the transmission of the packet is aborted. |
| 7:4 | CC       | Collision Count  
This 4-bit counter value indicates the number of collisions occurred before the packet was transmitted. The count is not valid when the EC bit is set. |
| 3   | ED       | Excessive Deferral  
This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times if DC bit is set in the Operating mode configuration register (ETH_MACCR). |
| 2   | UF       | Underflow Error  
This bit indicates that the MAC aborted the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions:  
The DMA encountered an empty Transmit Buffer while transmitting the packet  
The application filled the MTL Tx FIFO slower than the MAC transmit rate  
The transmission process enters the Suspend state and sets the underflow bit corresponding to a queue in the ETH_MTLISR register. |
| 1   | DB       | Deferred Bit  
This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the Half-duplex mode. |
| 0   | IHE      | IP Header Error  
When IP Header Error is set, this bit indicates that the Checksum Offload engine detected an IP header error. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload. |

1. This format is only applicable to the last descriptor of a packet.

Transmit context descriptor

The Transmit context descriptor can be provided any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor is used to provide the timestamps for one-step timestamp correction, and VLAN Tag ID for VLAN insertion feature. Write-back is only done on a context descriptor to reset the OWN bit.

Note: The VLAN tag IDs and MSS values, which are provided by the application in a context descriptor with their corresponding Valid bits set, are stored internally by the DMA. When the outer or inner VLAN tag is provided with the Valid bit set, the DMA always passes the last valid VLAN tag to the MTL. The application cannot invalidate the valid VLAN tag.
stored by the DMA. The VLAN tag is inserted or replaced based on the control inputs provided for the packet.

The Inner VLAN Tag Control input is used only for the packet that immediately follows the context descriptor. The application must provide a context descriptor before the normal descriptor of each packet for which the DMA should use the inner VLAN Tag control input.

Figure 824 shows the format for Transmit context descriptors. Table 661 to Table 664 provide a detailed description of all Transmit context descriptors.

Figure 824. Transmit context descriptor format

- TDES0 context descriptor (read format)

Table 661. TDES0 context descriptor

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>TTSL</td>
<td>Transmit Packet Timestamp Low</td>
</tr>
</tbody>
</table>

For one-step correction, the driver can provide the lower 32 bits of timestamp in this descriptor word. The DMA uses this value as the low word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

- TDES1 context descriptor (read format)
• TDES2 context descriptor (read format)

Table 663. TDES2 context descriptor

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16 IVT</td>
<td>Inner VLAN Tag</td>
</tr>
<tr>
<td>15:14</td>
<td>Reserved</td>
</tr>
<tr>
<td>13:0 MSS</td>
<td>Maximum Segment Size</td>
</tr>
</tbody>
</table>

- When the IVLTV bit of TDES3 context descriptor is set and the TCMSSV and OSTC bits of TDES3 context descriptor are reset, TDES2[31:16] contains the inner VLAN Tag to be inserted in the subsequent Transmit packets.
- This segment size is used while segmenting the TCP/IP payload. This field is valid only if the TCMSSV bit of TDES3 context descriptor is set and the OSTC bit of the TDES3 context descriptor is reset.

• TDES3 context descriptor (read format)

Table 664. TDES3 context descriptor

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 OWN</td>
<td>Own bit</td>
</tr>
<tr>
<td>30 CTXT</td>
<td>Context Type</td>
</tr>
<tr>
<td>29:28</td>
<td>Reserved</td>
</tr>
<tr>
<td>27 OSTC</td>
<td>One-Step Timestamp Correction Enable</td>
</tr>
</tbody>
</table>

- Own bit
  1: the DMA owns the descriptor.
  0: the application owns the descriptor.
  The DMA clears this bit immediately after a read operation.
- Context Type
  This bit should be set to 1 for context descriptor.
- Reserved
- When this bit is set, the DMA performs a one-step timestamp correction with reference to the timestamp values provided in TDES0 and TDES1.
One-Step Timestamp Correction Input or MSS Valid
When this bit and the OSTC bit are set, it indicates that the Timestamp Correction input provided in TDES0 and TDES1 is valid.
When the OSTC bit is reset and this bit and the TSE bit of TDES3 are set in subsequent normal descriptor, it indicates that the MSS input in TDES2 is valid.

Context Descriptor Error
When this bit is set, it indicates that the context descriptor is incorrect. The DMA sets this bit during write-back while closing the context descriptor.
The Context Descriptor errors can be:
– Incorrect sequence from the context descriptor. For example, a location before the first descriptor for a packet.
– All 1s.
– CD, LD, and FD bits set to 1.
Note: When a Context Descriptor error occurs due to All 1s or CTXT, LD, and FD bits set to 1, the Transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When IOC bit in TDES2 of corresponding first descriptor is set to 1, Transmit DMA sets the TI bit in the Channel status register (ETH_DMACSR).
Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the First Descriptor (even if FD bit is not set) and partial packet is sent.
This error is categorized as an abnormal event; recovery is only by issuing a software reset (DMA stopping-reconfiguring-restarting recovery mechanism is not supported)

Inner VLAN Tag Insert or Replace
When these bits are set, they request the MAC to perform Inner VLAN tagging or untagging before transmitting the packets. If the packet is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes.
This bitfield has the following values:
00: Do not add the inner VLAN tag.
01: Remove the inner VLAN tag from the packets before transmission. This option should be used only with the VLAN frames.
10: Insert an inner VLAN tag with the tag value programmed in the Inner VLAN inclusion register (ETH_MACIVIR) or context descriptor.
11: Replace the inner VLAN tag in packets with the tag value programmed in the Inner VLAN inclusion register (ETH_MACIVIR) or context descriptor. This option should be used only with the VLAN frames.

Inner VLAN Tag Valid
When this bit is set, it indicates that the IVT field of TDES2 is valid.
<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 16  | VLTV | VLAN Tag Valid  
When this bit is set, it indicates that the VT field of TDES3 is valid. |
| 15:0 | VT | VLAN Tag  
This field contains the VLAN Tag to be inserted or replaced in the packet. This field is used as VLAN Tag only when the VLTI bit of the VLAN inclusion register (ETH_MACVIR) is reset. |
57.10.4 Receive descriptor

The DMA in the Ethernet peripheral attempts to read a descriptor only if the Tail pointer is different from the Base pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC; otherwise, the performance of the DMA is greatly impacted because of the unavailability of the descriptors. In such a situation, the MTL RxFIFO becomes full and starts dropping packets.

The following Receive descriptors are present:

- Normal descriptors with read and write-back formats
- Context descriptors

All received descriptors are prepared by the software and given to the DMA as “normal” descriptors (see Figure 825: Receive normal descriptor (read format) for a description of their content). The DMA reads this descriptor and, after transferring a received packet (or part of it) to the buffers indicated by the descriptor, the Rx DMA closes the descriptor with the corresponding packet status. The status format is given in Figure 826: Receive normal descriptor (write-back format).

For some packets, the normal descriptor bits are not sufficient to write the complete status. For such packets, the Rx DMA writes the extended status to the next descriptor (without processing or using the Buffers pointers embedded in that descriptor). The format and content of this write-back descriptor is described in Figure 827: Receive context descriptor.

Receive normal descriptor (read format)

Figure 825 shows the read format for Receive normal descriptors. Table 665 to Table 668 provide a detailed description of all Receive normal descriptors (read format).

![Figure 825. Receive normal descriptor (read format)](image)

Note: In the Receive descriptor (read format), if the Buffer Address field contains only 0s, the MAC does not transfer data to this buffer and skips to the next buffer or next descriptor.
- **RDES0 normal descriptor (read format)**

  **Table 665. RDES0 normal descriptor (read format)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>BUF1AP</td>
<td><strong>Buffer 1 Address Pointer</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>These bits indicate the physical address of Buffer 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The application can program a byte-aligned address for this buffer, which</td>
</tr>
<tr>
<td></td>
<td></td>
<td>means that the LS bits of this field can be non-zero. However, while</td>
</tr>
<tr>
<td></td>
<td></td>
<td>transferring the start of packet, the DMA performs a write operation with</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RDES2[1:0]=0 in case of 64-/128-bit configuration) as zero. However, the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>packet data is shifted by the actual offset as given in the buffer address</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pointer. If the address pointer points to a buffer where the middle or last</td>
</tr>
<tr>
<td></td>
<td></td>
<td>part of the packet is stored, the DMA ignores the offset address and writes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to the full location as indicated by the data-width.</td>
</tr>
</tbody>
</table>

- **RDES1 normal descriptor (read format)**

  **Table 666. RDES1 normal descriptor (read format)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>Reserved</td>
<td>Field reserved.</td>
</tr>
</tbody>
</table>

- **RDES2 normal descriptor (read format)**

  **Table 667. RDES2 normal descriptor (read format)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>BUF2AP</td>
<td><strong>Buffer 2 Address Pointer</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>These bits indicate Buffer 2 physical address.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The RxDMA uses the LS bits of the pointer address only while transferring the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>start bytes of a packet. If the BUF2AP is giving the address of a buffer in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>which the middle or last part of a packet is stored, the DMA ignores RDES2[1:0]=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and writes to the complete location.</td>
</tr>
</tbody>
</table>

- **RDES3 normal descriptor (read format)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>OWN</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>IOC</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>BUF2V</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>BUF1V</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
### Table 668. RDES3 normal descriptor (read format)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>OWN</td>
<td>Own bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– The DMA completes the packet reception</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– The buffers associated with the descriptor are full</td>
</tr>
<tr>
<td>30</td>
<td>IOC</td>
<td>Interrupt Enabled on Completion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.</td>
</tr>
<tr>
<td>29:26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>BUF2V</td>
<td>Buffer 2 Address Valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, it indicates to the DMA that the buffer 2 address specified in RDES2 is valid. The application must set this bit so that the DMA can use the address to which the Buffer 2 address in RDES0 is pointing, to write received packet data.</td>
</tr>
<tr>
<td>24</td>
<td>BUF1V</td>
<td>Buffer 1 Address Valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When set, this indicates to the DMA that the buffer 1 address specified in RDES0 is valid. The application must set this value if the address to which Buffer 1 address points in RDES0, can be used by the DMA to write received packet data.</td>
</tr>
<tr>
<td>23:0</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
Receive normal descriptor (write-back format)

Figure 826 shows the write-back format for Receive normal descriptors. Table 669 to Table 672 provide a detailed description of all Receive normal descriptors (write-back format).

![Figure 826. Receive normal descriptor (write-back format)](image)

- RDES0 normal descriptor (write-back format)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>IVT</td>
<td>Inner VLAN Tag This field contains the Inner VLAN tag of the received packet if the RS0V bit of RDES3 is set. This is valid only when Double VLAN tag processing and VLAN tag stripping are enabled.</td>
</tr>
<tr>
<td>15:0</td>
<td>OVT</td>
<td>Outer VLAN Tag This field contains the Outer VLAN tag of the received packet if the RS0V bit of RDES3 is set.</td>
</tr>
</tbody>
</table>

- RDES1 normal descriptor (write-back format)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TD</td>
<td></td>
</tr>
<tr>
<td>TSA</td>
<td></td>
</tr>
<tr>
<td>PV</td>
<td></td>
</tr>
<tr>
<td>PFT</td>
<td></td>
</tr>
<tr>
<td>PMT</td>
<td></td>
</tr>
<tr>
<td>IPCE</td>
<td></td>
</tr>
<tr>
<td>IPCB</td>
<td></td>
</tr>
<tr>
<td>IPV6</td>
<td></td>
</tr>
<tr>
<td>IPV4</td>
<td></td>
</tr>
<tr>
<td>IPHE</td>
<td></td>
</tr>
<tr>
<td>PT</td>
<td></td>
</tr>
</tbody>
</table>
Table 670. RDES1 normal descriptor (write-back format)\(^{(1)}\)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>OPC</td>
<td><strong>OAM Subtype Code, or MAC Control Packet opcode</strong>&lt;br&gt; OAM Subtype Code&lt;br&gt; If bits[18:16] of RDES3 are set to 111, this field contains the OAM subtype and code fields.&lt;br&gt; MAC Control Packet opcode&lt;br&gt; If bits[18:16] of RDES3 are set to 110, this field contains the MAC Control packet opcode field.</td>
</tr>
<tr>
<td>15</td>
<td>TD</td>
<td><strong>Timestamp Dropped</strong>&lt;br&gt; This bit indicates that the timestamp was captured for this packet but got dropped in the MTL Rx FIFO because of overflow.</td>
</tr>
<tr>
<td>14</td>
<td>TSA</td>
<td><strong>Timestamp Available</strong>&lt;br&gt; When Timestamp is present, this bit indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the Last Descriptor bit (RDES3 [28]) is set.&lt;br&gt; The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.</td>
</tr>
<tr>
<td>13</td>
<td>PV</td>
<td><strong>PTP Version</strong>&lt;br&gt; 1: Received PTP message in IEEE 1588 version 2 format&lt;br&gt; 0: Received PTP message in IEEE 1588 version 1 format</td>
</tr>
<tr>
<td>12</td>
<td>PFT</td>
<td><strong>PTP Packet Type</strong>&lt;br&gt; This bit indicates that the PTP message is sent directly over Ethernet.</td>
</tr>
<tr>
<td>11:8</td>
<td>PMT</td>
<td><strong>PTP Message Type</strong>&lt;br&gt; These bits are encoded to give the type of the message received:&lt;br&gt; 0000: No PTP message received&lt;br&gt; 0001: SYNC (all clock types)&lt;br&gt; 0010: Follow_Up (all clock types)&lt;br&gt; 0011: Delay_Req (all clock types)&lt;br&gt; 0100: Delay_Resp (all clock types)&lt;br&gt; 0101: Pdelay_Req (in peer-to-peer transparent clock)&lt;br&gt; 0110: Pdelay_Resp (in peer-to-peer transparent clock)&lt;br&gt; 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock)&lt;br&gt; 1000: Announce&lt;br&gt; 1001: Management&lt;br&gt; 1010: Signaling&lt;br&gt; 1011–1110: Reserved&lt;br&gt; 1111: PTP packet with Reserved message type</td>
</tr>
<tr>
<td>7</td>
<td>IPCE</td>
<td><strong>IP Payload Error</strong>&lt;br&gt; When this bit is set, it indicates either of the following:&lt;br&gt; – The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment.&lt;br&gt; – The TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field.&lt;br&gt; – The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP.&lt;br&gt; Bit 15 (ES) of RDES3 is not set when this bit is set.</td>
</tr>
</tbody>
</table>
### Table 670. RDES1 normal descriptor (write-back format)<sup>(1)</sup> (continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 6   | IPCB  | IP Checksum Bypassed  
This bit indicates that the checksum offload engine is bypassed. |
| 5   | IPV6  | IPv6 header Present  
This bit indicates that an IPv6 header is detected. |
| 4   | IPV4  | IPv4 Header Present  
This bit indicates that an IPv4 header is detected. |
| 3   | IPHE  | IP Header Error  
– When this bit is set, it indicates either of the following:  
  – The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes.  
  – The IP datagram version is not consistent with the Ethernet Type value.  
  – Ethernet packet does not have the expected number of IP header bytes.  
This bit is valid when either bit 5 or bit 4 is set. |
| 2:0 | PT    | Payload Type  
These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE):  
  000: Unknown type or IP/AV payload not processed  
  001: UDP  
  010: TCP  
  011: ICMP  
  100: IGMP if IPV4 Header Present bit is set  
Others: reserved.  
If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 3'b000. |

1. The Status fields in write-back format are valid only for the last descriptor (RDES3[28] is set).

- **RDES2 normal descriptor (write-back format)**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3L4FM</td>
<td>L4FM</td>
<td>L3FM</td>
<td>MADRM</td>
<td>HF</td>
<td>DAF</td>
<td>SAF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>VF</td>
<td>Reserved</td>
<td>ARPRN</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 671. RDES2 normal descriptor (write-back format)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:29 | L3L4FM | **Layer 3 and Layer 4 Filter Number Match**  
These bits indicate the number of the Layer 3 and Layer 4 Filter that matched the received packet:  
– 000: Filter 0  
– 001: Filter 1  
– 010: Filter 2  
– 011: Filter 3  
– 100: Filter 4  
– 101: Filter 5  
– 110: Filter 6  
– 111: Filter 7  
This field is valid only when bit 28 or bit 27 is set high. When more than one filter matches, these bits give the number of lowest filter. |
| 28    | L4FM   | **Layer 4 Filter Match**  
When this bit is set, it indicates that the received packet matches one of the enabled Layer 4 Port Number fields. This status is given only when one of the following conditions is true:  
– Layer 3 fields are not enabled and all enabled Layer 4 fields match  
– All enabled Layer 3 and Layer 4 filter fields match  
When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by bits[31:29]. |
| 27    | L3FM   | **Layer 3 Filter Match**  
When this bit is set, it indicates that the received packet matches one of the enabled Layer 3 IP Address fields. This status is given only when one of the following conditions is true:  
– All enabled Layer 3 fields match and all enabled Layer 4 fields are bypassed  
– All enabled filter fields match  
When more than one filter matches, this bit gives the layer 3 filter status of filter indicated by bits[31:29]. |
| 26:19 | MADRM  | **MAC Address Match or Hash Value**  
When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset.  
When the HF bit is set, this field contains the Hash value computed by the MAC. A packet passes the Hash filter when the bit corresponding to the Hash value is set in the Hash filter register. |
| 18    | HF     | **Hash Filter Status**  
When this bit is set, it indicates that the packet passed the MAC address Hash filter. Its[26:19] indicate the Hash value. |
| 17    | DAF    | **Destination Address Filter Fail**  
When this bit is set, it indicates that the packet failed the DA Filter in the MAC. |
| 16    | SAF    | **SA Address Filter Fail**  
When this bit is set, it indicates that the packet failed the SA Filter in the MAC. |
Table 671. RDES2 normal descriptor (write-back format) (continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 15  | VF   | VLAN Filter Status  
When this bit is set, it indicates that the VLAN Tag of received packet passed the VLAN filter. |
| 14:11 | Reserved | |
| 10  | ARPNR | ARP Reply Not Generated  
When this bit is set, it indicates that the MAC did not generate the ARP Reply for received ARP Request packet. This bit is set when the MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time). |
| 9:0  | Reserved | |

- **RDES3 normal descriptor (write-back format)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31  | OWN  | Own bit  
1: The DMA owns the descriptor.  
0: The application owns the descriptor.  
The DMA clears this bit when either of the following conditions is true:  
The DMA completes the packet reception  
The buffers associated with the descriptor are full |
| 30  | CTXT | Receive Context Descriptor  
When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 0 to this bit for normal receive descriptor.  
When CTXT and FD bits are used together, {CTXT, FD} possible values are:  
00: Intermediate Descriptor  
01: First Descriptor  
10: Reserved  
11: Descriptor error (due to all 1s)  
*Note: When a Descriptor error occurs, the Receive DMA closes the receive descriptor indicating a Descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data. The receive DMA sets the CDE field of the Channel status register (ETH_DMACSR) but does not set the RI field, even when IOC field is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.* |

Table 672. RDES3 normal descriptor (write-back format)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31  | OWN  | Own bit  
1: The DMA owns the descriptor.  
0: The application owns the descriptor.  
The DMA clears this bit when either of the following conditions is true:  
The DMA completes the packet reception  
The buffers associated with the descriptor are full |
| 30  | CTXT | Receive Context Descriptor  
When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 0 to this bit for normal receive descriptor.  
When CTXT and FD bits are used together, {CTXT, FD} possible values are:  
00: Intermediate Descriptor  
01: First Descriptor  
10: Reserved  
11: Descriptor error (due to all 1s)  
*Note: When a Descriptor error occurs, the Receive DMA closes the receive descriptor indicating a Descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data. The receive DMA sets the CDE field of the Channel status register (ETH_DMACSR) but does not set the RI field, even when IOC field is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.* |
Table 672. RDES3 normal descriptor (write-back format) (continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>FD</td>
<td><strong>First Descriptor</strong>&lt;br&gt;When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet. Refer to the CTXT bit description for details on how to use the CTXT bit and FD bit together.</td>
</tr>
<tr>
<td>28</td>
<td>LD</td>
<td><strong>Last Descriptor</strong>&lt;br&gt;When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.</td>
</tr>
<tr>
<td>27</td>
<td>RS2V</td>
<td><strong>Receive Status RDES2 Valid</strong>&lt;br&gt;When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.</td>
</tr>
<tr>
<td>26</td>
<td>RS1V</td>
<td><strong>Receive Status RDES1 Valid</strong>&lt;br&gt;When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.</td>
</tr>
<tr>
<td>25</td>
<td>RS0V</td>
<td><strong>Receive Status RDES0 Valid</strong>&lt;br&gt;When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.</td>
</tr>
<tr>
<td>24</td>
<td>CE</td>
<td><strong>CRC Error</strong>&lt;br&gt;When this bit is set, it indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.</td>
</tr>
<tr>
<td>23</td>
<td>GP</td>
<td><strong>Giant Packet</strong>&lt;br&gt;When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable is set).&lt;br&gt;Giant packet indicates only the packet length. It does not cause any packet truncation.</td>
</tr>
<tr>
<td>22</td>
<td>RWT</td>
<td><strong>Receive Watchdog Timeout</strong>&lt;br&gt;When this bit is set, it indicates that the Receive Watchdog Timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.</td>
</tr>
<tr>
<td>21</td>
<td>OE</td>
<td><strong>Overflow Error</strong>&lt;br&gt;When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO.&lt;br&gt;This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial packets are dropped completely in Rx FIFO.</td>
</tr>
<tr>
<td>20</td>
<td>RE</td>
<td><strong>Receive Error</strong>&lt;br&gt;When this bit is set, it indicates that the ETH_RX_ER signal is asserted while the ETH_RX_DV signal is asserted during packet reception.</td>
</tr>
</tbody>
</table>
### Table 672. RDES3 normal descriptor (write-back format) (continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 19  | DE   | Dribble Bit Error  
When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode. |
| 18:16 | LT | Length/Type Field  
This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows:  
000: The packet is a length packet  
001: The packet is a type packet.  
011: The packet is a ARP Request packet type  
100: The packet is a type packet with VLAN Tag  
101: The packet is a type packet with Double VLAN Tag  
110: The packet is a MAC Control packet type  
111: The packet is a OAM packet type  
010: Reserved |
| 15  | ES   | Error Summary  
When this bit is set, it indicates the logical OR of the following bits:  
RDES3[19]: Dribble Error  
RDES3[20]: Receive Error  
RDES3[21]: Overflow Error  
RDES3[22]: Watchdog Timeout  
RDES3[23]: Giant Packet  
RDES3[24]: CRC Error  
This field is valid only when the LD bit of RDES3 is set. |
| 14:0 | PL  | Packet Length  
These bits indicate the byte length of the received packet that was transferred to system memory (including CRC).  
This field is valid when both the LD bit of RDES3 is set and the Overflow Error bit is reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet.  
When LD bit of RDES3 is reset, this field contains the accumulated number of bytes (partial) that have been transferred for the current packet. |
Receive context descriptor

This descriptor is read-only for the application. This descriptor can be written only by the DMA.

The context descriptor provides information about the extended status related to the last received packet. Bit 30 of RDES3 indicates the context type descriptor.

*Figure 827* shows the format for Receive context descriptors. *Table 673* to *Table 676* provide a detailed description of all Receive context descriptors.

*Figure 827. Receive context descriptor*

- RDES0 context descriptor

*Table 673. RDES0 context descriptor*

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | RTSL | Receive Packet Timestamp Low  
The DMA updates this field with least significant 32 bits of the timestamp captured for corresponding Receive packet. When this field and the RTSH field of RDES1 show all-ones value, the timestamp must be considered as corrupt. |
• RDES1 context descriptor

Table 674. RDES1 context descriptor

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>RTSH</td>
<td>Receive Packet Timestamp High</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The DMA updates this field with most significant 32 bits of the timestamp captured for corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, the timestamp must be considered as corrupt.</td>
</tr>
</tbody>
</table>

• RDES2 context descriptor

Table 675. RDES2 context descriptor

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

• RDES3 context descriptor

Table 676. RDES3 context descriptor

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>OWN</td>
<td>Own Bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: The DMA owns the descriptor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: The application owns the descriptor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The DMA clears this bit when either of the following conditions is true:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The DMA completes the packet reception</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The buffers associated with the descriptor are full</td>
</tr>
<tr>
<td>30</td>
<td>CTXT</td>
<td>Receive Context Descriptor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, it indicates that the current descriptor is a context descriptor. The DMA writes 1'b1 to this bit for context descriptor.</td>
</tr>
<tr>
<td>29:0</td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>
57.11 Ethernet registers

57.11.1 Ethernet register maps

This section provides the following register maps:

- DMA registers (see Section 57.11.2: Ethernet DMA registers)
- MTL registers (see Section 57.11.3: Ethernet MTL registers)
- MAC registers including MMC register (see Section 57.11.4: Ethernet MAC and MMC registers)

57.11.2 Ethernet DMA registers

DMA mode register (ETH_DMAMR)

Address offset: 0x1000
Reset value: 0x0000 0000

The DMA mode register establishes the bus operating modes for the DMA.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res.</td>
<td>PR[2:0]</td>
<td>TXPR</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 INTM[1:0]: Interrupt Mode

This field defines the interrupt mode of the Ethernet peripheral.
The behavior of the interrupt signal and of the RI/TI bits in the ETH_DMACSR register changes depending on the INTM value (refer to Table 652: Transfer complete interrupt behavior).

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 PR[2:0]: Priority ratio

These bits control the priority ratio in weighted round-robin arbitration between the Rx DMA and Tx DMA. These bits are valid only when the DA bit is reset. The priority ratio is Rx:Tx or Tx:Rx depending on whether the TXPR bit is reset or set.

- 000: The priority ratio is 1:1
- 001: The priority ratio is 2:1
- 010: The priority ratio is 3:1
- 011: The priority ratio is 4:1
- 100: The priority ratio is 5:1
- 101: The priority ratio is 6:1
- 110: The priority ratio is 7:1
- 111: The priority ratio is 8:1

Bit 11 TXPR: Transmit priority

When set, this bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus.
Bits 10:2  Reserved, must be kept at reset value.

Bit 1  **DA**: DMA Tx or Rx Arbitration Scheme

This bit specifies the arbitration scheme between the Transmit and Receive paths of all channels:

0: Weighted Round-Robin with Rx:Tx or Tx:Rx

The priority between the paths is according to the priority specified in Bits[14:12] and the priority weight is specified in the TXPR bit.

1: Fixed priority

The Tx path has priority over the Rx path when the TXPR bit is set. Otherwise, the Rx path has priority over the Tx path.

Bit 0  **SWR**: Software Reset

When this bit is set, the MAC and the DMA controller reset the logic and all internal registers of the DMA, MTL, and MAC. This bit is automatically cleared after the reset operation is complete in all clock domains. Before reprogramming any register, a value of zero should be read in this bit.

**Note:** The reset operation is complete only when all resets in all active clock domains are deasserted. Therefore, it is essential that all PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion. The time to complete the software reset operation depends on the frequency of the slowest active clock.
System bus mode register (ETH_DMASBMR)

Address offset: 0x1004
Reset value: 0x0000 0000

The System bus mode register controls the behavior of the AHB master. It mainly controls burst splitting and number of outstanding requests.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>rw</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **RB**: Rebuild INCRx Burst
When this bit is set high and the AHB master gets SPLIT, RETRY, or Early Burst Termination (EBT) response, the AHB master interface rebuilds the pending beats of any initiated burst transfer with INCRx and SINGLE transfers. By default, the AHB master interface rebuilds pending beats of an EBT with an unspecified (INCR) burst.

Bit 14 **MB**: Mixed Burst
When this bit is set high and the FB bit is low, the AHB master performs undefined bursts transfers (INCR) for burst length of 16 or more. For burst length of 16 or less, the AHB master performs fixed burst transfers (INCRx and SINGLE).

Bit 13 Reserved, must be kept at reset value.

Bit 12 **AAL**: Address-Aligned Beats
When this bit is set to 1, the master performs address-aligned burst transfers on Read and Write channels.

Bits 11:1 Reserved, must be kept at reset value.

Bit 0 **FB**: Fixed Burst Length
When this bit is set to 1, the AHB master will initiate burst transfers of specified length (INCRx or SINGLE).
When this bit is set to 0, the AHB master will initiate transfers of unspecified length (INCR) or SINGLE transfers.
**Interrupt status register (ETH_DMAISR)**

Address offset: 0x1008  
Reset value: 0x0000 0000  
The application reads this Interrupt Status register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>MACIS</td>
<td>MAC Interrupt Status</td>
</tr>
<tr>
<td>30</td>
<td>MTLIS</td>
<td>MTL Interrupt Status</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>DC0IS</td>
<td>DMA Channel Interrupt Status</td>
</tr>
</tbody>
</table>

Bits 31:18 Reserved, must be kept at reset value.

- **Bit 17 MACIS**: MAC Interrupt Status  
  This bit indicates an interrupt event in the MAC. To reset this bit to 1'b0, the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.

- **Bit 16 MTLIS**: MTL Interrupt Status  
  This bit indicates an interrupt event in the MTL. To reset this bit to 1'b0, the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source.

Bits 15:1 Reserved, must be kept at reset value.

- **Bit 0 DC0IS**: DMA Channel Interrupt Status  
  This bit indicates an interrupt event in DMA Channel. To reset this bit to 0, the software must read the corresponding register in DMA Channel to get the exact cause of the interrupt and clear its source.

**Debug status register (ETH_DMADSR)**

Address offset: 0x100C  
Reset value: 0x0000 0000  
The Debug status register gives the Receive and Transmit process status for DMA Channel for debugging purpose.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>TPS0[3:0]</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>RPS0[3:0]</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>AXWH</td>
<td></td>
</tr>
</tbody>
</table>

2810/3119  
RM0481 Rev 1
Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **TPS0[3:0]**: DMA Channel Transmit Process State

This field indicates the Tx DMA FSM state for Channel:

- 000: Stopped (Reset or Stop Transmit Command issued)
- 001: Running (Fetching Tx Transfer Descriptor)
- 010: Running (Waiting for status)
- 011: Running (Reading Data from system memory buffer and queuing it to the Tx buffer (Tx FIFO))
- 100: Timestamp write state
- 101: Reserved for future use
- 110: Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
- 111: Running (Closing Tx Descriptor)

The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 11:8 **RPS0[3:0]**: DMA Channel Receive Process State

This field indicates the Rx DMA FSM state for Channel:

- 000: Stopped (Reset or Stop Receive Command issued)
- 001: Running (Fetching Rx Transfer Descriptor)
- 010: Reserved for future use
- 011: Running (Waiting for Rx packet)
- 100: Suspended (Rx Descriptor Unavailable)
- 101: Running (Closing the Rx Descriptor)
- 110: Timestamp write state
- 111: Running (Transferring the received packet data from the Rx buffer to the system memory)

The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **AXWHSTS**: AHB Master Write Channel

When high, this bit indicates that the write channel of the AHB master FMSs are in non-idle state.

**Channel control register (ETH_DMACCR)**

Address offset: 0x1100

Reset value: 0x0000 0000

The DMA Channel control register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.
Bits 31:21  Reserved, must be kept at reset value.

Bits 20:18  **DSL[2:0]: Descriptor Skip Length**
This bit specifies the 32-bit word number to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor.
When the DSL value is equal to zero, the DMA takes the descriptor table as contiguous.

Bit 17   Reserved, must be kept at reset value.

Bit 16  **PBLX8: 8xPBL mode**
When this bit is set, the PBL value programmed in Bits[21:16] in *Channel transmit control register (ETH_DMACTXCR)* is multiplied eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.

Bits 15:14   Reserved, must be kept at reset value.

Bits 13:0  **MSS[13:0]: Maximum Segment Size**
This field specifies the maximum segment size that should be used while segmenting the packet. This field is valid only if the TSE bit of *Channel transmit control register (ETH_DMACTXCR)* is set.
The value programmed in this field must be more than the configured Data width in bytes. It is recommended to use a MSS value of 64 bytes or more.

### Channel transmit control register (ETH_DMACTXCR)

Address offset: 0x1104

Reset value: 0x0000 0000

The DMA Channel Transmit Control register controls the Tx features such as PBL, TCP segmentation, and Tx Channel weights.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>ST</td>
</tr>
</tbody>
</table>
Bits 31:22  Reserved, must be kept at reset value.

Bits 21:16  **TXPBL[5:0]: Transmit Programmable Burst Length**

These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 beats, perform the following steps:

a)  Set the PBLx8 mode in ETH_DMACCR.

b)  Set the TXPBL[5:0].

Note: The maximum value of TXPBL must be less than or equal to half the Tx Queue size (TQS field of *Tx queue operating mode Register (ETH_MTLTXQOMR)*) in terms of beats. This is required so that the Tx Queue has space to store at least another Tx PBL worth of data while the MTL Tx Queue Controller is transferring data to MAC. The total locations in Tx Queue of size 2048 bytes is 512, TXPBL and 8xPBL needs to be programmed to less than or equal to 512/2.

Bits 15:13  Reserved, must be kept at reset value.

Bit 12  **TSE: TCP Segmentation Enabled**

When this bit is set, the DMA performs the TCP segmentation for packets in Channel x. The TCP segmentation is done only for those packets for which the TSE bit (TDES0[19]) is set in the Tx Normal descriptor. When this bit is set, the TxPBL value must be greater than or equal to 4.

Bits 11:5  Reserved, must be kept at reset value.

Bit 4  **OSF: Operate on Second Packet**

When this bit is set, it instructs the DMA to process the second packet of the Transmit data even before the status for the first packet is obtained.

Bits 3:1  Reserved, must be kept at reset value.

Bit 0  **ST: Start or Stop Transmission Command**

When this bit is set, transmission is placed in the Running state. The DMA checks the Transmit list at the current position for a packet to be transmitted. The DMA tries to acquire descriptor from either of the following positions:

-  The current position in the list: this is the base address of the Transmit list set by the ETH_DMACTXDLAR register.

-  The position at which the transmission was previously stopped

If the DMA does not own the current descriptor, the transmission enters the Suspended state and the TBU bit of the ETH_DMACSR is set. The Start Transmission command is effective only when the transmission is stopped. If the command is issued before setting the ETH_DMACTXDLAR register, the DMA behavior is unpredictable.

When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current packet. The Next Descriptor position in the Transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, you need to program ETH_DMACTXDLAR register with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the Suspended state.
Channel receive control register (ETH_DMACRXCR)

Address offset: 0x1108
Reset value: 0x0000 0000

The DMA Channel Receive Control register controls the Rx features such as PBL, buffer size, and extended status.

Bit 31 **RPF**: DMA Rx Channel Packet Flush
When this bit is set to 1, the Ethernet peripheral automatically flushes the packet from the Rx queues destined to DMA Rx Channel when the DMA Rx Channel is stopped after a system bus error has occurred. When this bit remains set and the DMA is re-started by the software driver, the packets residing in the Rx Queues that were received when this RxDMA was stopped, are flushed out. The packets that are received by the MAC after the RxDMA is re-started are routed to the RxDMA. The flushing happens on the Read side of the Rx queue. When this bit is set to 0 the Ethernet peripheral does not flush the packet in the Rx queue destined to DMA Rx Channel after the DMA is stopped due to a system bus error. This might cause head-of-line blocking in the corresponding RxQueue.

*Note*: The stopping of packet flow from a Rx DMA Channel to the application by setting RPF works only when there is one-to-one mapping of Rx Queue to Rx DMA channels. In Dynamic mapping mode, setting RPF bit in ETH_DMACRXCR register might flush packets from unintended Rx Queues which are destined to the stopped Rx DMA Channel.

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:16 **RXPBL[5:0]**: Receive Programmable Burst Length
These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 beats, perform the following steps:

a) Set the PBLx8 mode in the ETH_DMACCCR.
b) Set the RXPBL[5:0].

*Note*: The maximum value of RXPBL must be less than or equal to half the Rx Queue size (RQS field of Rx queue operating mode register (ETH_MTLRXQOMR)) in terms of beats. This is required so that the Rx Queue has space to store at least another Rx PBL worth of data while the MTL Rx Queues Controller is transferring data to MAC. The total locations in Rx Queue of size 2048 bytes is 512, RXPBL and 8xPBL needs to be programmed to less than or equal to 512/2.
Bit 15  Reserved, must be kept at reset value.

Bits 14:1  **RBSZ[13:0]:** Receive Buffer size

This field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16 Kbytes.

**Note:** The buffer size must be a multiple of 4. This is required even if the value of buffer address pointer is not aligned to bus width. If the buffer size is not a multiple of 4, it may result into an undefined behavior.

The LSB bits (1:0) are ignored and the DMA internally takes the LSB bits as all-zero. Therefore, these LSB bits are read-only (RO).

Bit 0  **SR:** Start or Stop Receive

When this bit is set, the DMA tries to acquire the descriptor from the Receive list and processes the incoming packets.

The DMA tries to acquire descriptor from either of the following positions:

- The current position in the list: this is the address set by the Channel Rx descriptor list address register (ETH_DMACRXDLAR).
- The position at which the Rx process was previously stopped

If the DMA does not own the current descriptor, the reception is suspended and the RBU bit of the ETH_DMACSR is set. The Start Receive command is effective only when the reception is stopped. If the command is issued before setting the Channel Rx descriptor list address register (ETH_DMACRXDLAR), the DMA behavior is unpredictable.

When this bit is reset, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the Receive list is saved, and it becomes the current position after the Rx process is restarted. The Stop Receive command is effective only when the Rx process is in the Running (waiting for Rx packet) or Suspended state.
Channel Tx descriptor list address register (ETH_DMACTXDLAR)

Address offset: 0x1114
Reset value: 0x0000 0000

Channel Tx Descriptor List Address register points the DMA to the start of Transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be word-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

You can write to this register only when the Tx DMA has stopped, that is, the ST bit is set to zero in ETH_DMACTXCR register. When stopped, this register can be written with a new descriptor list address. When you set ST bit to 1, the DMA takes the newly-programmed descriptor base address. If this register is not changed when ST bit is set to 0, the DMA takes the descriptor address where it was stopped earlier.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

TDESLA[31:16]  
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

TDESLA[15:0]  
rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw

Bits 31:0  TDESLA[31:0]: Start of Transmit List
This field contains the base address of the first descriptor in the Transmit descriptor list. The DMA ignores the LSB bits (1:0) for 32-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).
Channel Rx descriptor list address register (ETH_DMACRXDLAR)

Address offset: 0xF1C
Reset value: 0x0000 0000

The Channel Rx Descriptor List Address register points the DMA to the start of Receive descriptor list.

This register points to the start of the Receive Descriptor List. The descriptor lists reside in the physical memory space of the application and must be word-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, SR bit is set to zero in ETH_DMACRXCR register. When stopped, this register can be written with a new descriptor list address.

When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **RDESLA[31:0]**: Start of Receive List
This field contains the base address of the first descriptor in the Rx Descriptor list. The DMA ignores the LSB bits (1:0) for 32-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).
Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR)

Address offset: 0x1120
Reset value: 0x0000 0000

The Channel Tx Descriptor Tail Pointer register points to an offset from the base and indicates the location of the last valid descriptor.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>TDT[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>TDT[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **TDT[31:0]**: Transmit Descriptor Tail Pointer
This field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers.
Channel Rx descriptor tail pointer register (ETH_DMACRXDTPR)

Address offset: 0x1128
Reset value: 0x0000 0000

The Channel Rx Descriptor Tail Pointer Points to an offset from the base and indicates the location of the last valid descriptor.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 RDT[31:0]: Receive Descriptor Tail Pointer
This field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.

Channel Tx descriptor ring length register (ETH_DMACTXRLR)

Address offset: 0x112C
Reset value: 0x0000 0000

The Tx Descriptor Ring Length register contains the length of the Transmit descriptor ring.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 TDRL[9:0]: Transmit Descriptor Ring Length
This field sets the maximum number of Tx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. It is recommended to put a minimum ring descriptor length of 4.

For example, you can program any value up to 0x3FF in this field. This field is 10 bits wide, if you program 0x3FF, you can have 1024 descriptors. If you want to have 10 descriptors, program it to a value of 0x9.
Channel Rx descriptor ring length register (ETH_DMACRXRLR)

Address offset: 0x1130
Reset value: 0x0000 0000

The Channel Rx Descriptor Ring Length register contains the length of the Receive descriptor circular ring.

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>ARBS[7:0]</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Bits 31:24  Reserved, must be kept at reset value.
Bits 23:16  **ARBS[7:0]: Alternate Receive Buffer Size**
Indicates size in bytes for Buffer 1 when ARBS[7:0] is programmed to a non-zero value.
When ARBS[7:0] = 0, Rx Buffer1 and Rx Buffer2 sizes are based on RBSZ[13:0] field of
Channel receive control register (ETH_DMACRXCR).

Bits 15:10  Reserved, must be kept at reset value.
Bits 9:0   **RDRL[9:0]: Receive Descriptor Ring Length**
This register sets the maximum number of Rx descriptors in the circular descriptor ring. The
maximum number of descriptors is limited to 1K descriptors.
For example, you can program any value up to 0x3FF in this field. This field is 10-bit wide. If
you program 0x3FF, you can have 1024 descriptors. If you want to have 10 descriptors,
program it to a value of 0x9.

Channel interrupt enable register (ETH_DMACIER)

Address offset: 0x1134
Reset value: 0x0000 0000

The Channel Interrupt Enable register enables the interrupts reported by the Status register.

```
|    |    |    | NIE| AIE| CDEE| FBEE| ERIE| ETIE| RWTE| RSTE| R8UE| RIEE| RSE| Res| Res| TBUET| TXS| TIE |
|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|----|----|----|------|----|-----|
| 31 | 30 | 29 |    |    |    |    |    |    |    |    |    |    |    |    |    |     |    |    |
| 15 | 14 | 13 | 12 | 11 | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |     |    |    |
```
Bits 31:16  Reserved, must be kept at reset value.

Bit 15 **NIE**: Normal Interrupt Summary Enable
When this bit is set, the normal interrupt summary is enabled. This bit enables the following interrupts in the Channel status register (ETH_DMACSR):
- Bit 0: Transmit Interrupt
- Bit 2: Transmit Buffer Unavailable
- Bit 6: Receive Interrupt
- Bit 11: Early Receive Interrupt
When this bit is reset, the normal interrupt summary is disabled.

Bit 14 **AIE**: Abnormal Interrupt Summary Enable
When this bit is set, the abnormal interrupt summary is enabled. This bit enables the following interrupts in the Channel status register (ETH_DMACSR):
- Bit 1: Transmit Process Stopped
- Bit 7: Rx Buffer Unavailable
- Bit 8: Receive Process Stopped
- Bit 9: Receive Watchdog Timeout
- Bit 10: Early Transmit Interrupt
- Bit 12: Fatal Bus Error
When this bit is reset, the abnormal interrupt summary is disabled.

Bit 13 **CDEE**: Context Descriptor Error Enable
When this bit is set along with the AIE bit, the Context Descriptor error interrupt is enabled. When this bit is reset, the Context Descriptor error interrupt is disabled.

Bit 12 **FBEE**: Fatal Bus Error Enable
When this bit is set along with the AIE bit, the Fatal Bus error interrupt is enabled. When this bit is reset, the Fatal Bus Error error interrupt is disabled.

Bit 11 **ERIE**: Early Receive Interrupt Enable
When this bit is set along with the NIE bit, the Early Receive interrupt is enabled. When this bit is reset, the Early Receive interrupt is disabled.

Bit 10 **ETIE**: Early Transmit Interrupt Enable
When this bit is set along with the AIE bit, the Early Transmit interrupt is enabled. When this bit is reset, the Early Transmit interrupt is disabled.

Bit 9 **RWTE**: Receive Watchdog Timeout Enable
When this bit is set along with the AIE bit, the Receive Watchdog Timeout interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout interrupt is disabled.

Bit 8 **RSE**: Receive Stopped Enable
When this bit is set along with the AIE bit, the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped interrupt is disabled.

Bit 7 **RBUE**: Receive Buffer Unavailable Enable
When this bit is set along with the AIE bit, the Receive Buffer Unavailable interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable interrupt is disabled.

Bit 6 **RIE**: Receive Interrupt Enable
When this bit is set along with the NIE bit, the Receive Interrupt is enabled. When this bit is reset, the Receive Interrupt is disabled.

Bits 5:3  Reserved, must be kept at reset value.
Bit 2  **TBUE**: Transmit Buffer Unavailable Enable
   When this bit is set along with the NIE bit, the Transmit Buffer Unavailable interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable interrupt is disabled.

Bit 1  **TXSE**: Transmit Stopped Enable
   When this bit is set along with the AIE bit, the Transmission Stopped interrupt is enabled.
   When this bit is reset, the Transmission Stopped interrupt is disabled.

Bit 0  **TIE**: Transmit Interrupt Enable
   When this bit is set along with the NIE bit, the Transmit Interrupt is enabled. When this bit is reset, the Transmit Interrupt is disabled.
Figure 828. Generation of ETH_DMAISR flags

- Q#RXO
- Q#RXOIE
- Q#TXU
- Q#TXUIE
- OR
- MTLIS

- AND
- OR
- MACIS
- OR
- sbd_intr_o

- MMCRXIS
- MMCTXIS
- RXSTSIS
- RXSTSIIE
- TXSTSIS
- TXSTSIIE
- TSIS
- TSIE
- OR
- AND
- LPIIS
- LPIIE
- PMTSIS
- PMTSIE
- PHYIS
- PHYIE
- OR
- AND
- PCSANCIS
- PCSANCIE
- PCSLCHGIS
- PCSLCHGIE
- RGSMIIIS
- RGSMIIIE
- AND

- OR
- AND
- TI
- TIE
- ERI
- ERE
- TPS
- TSE
- FBI
- FBE
- AND
- OR
- NIS
- NIE
- AND
- OR
- AIS
- AIE
- AND

- DC#IS

MSv40824V2
Channel Rx interrupt watchdog timer register (ETH_DMACRXIWTR)

Address offset: 0x1138
Reset value: 0x0000 0000

The Receive Interrupt Watchdog Timer register indicates the watchdog timeout for Receive Interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the RI bit of the Channel status register (ETH_DMACSR).

### Register Layout

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Description</th>
<th>Range</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:18</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17:16</td>
<td>RWTU[1:0]: Receive Interrupt Watchdog Timer Count Units</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This field indicates the number of system clock cycles corresponding to one unit in RWT[7:0] field.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00: 256</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01: 512</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10: 1024</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11: 2048</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For example, when RWT[7:0] = 2 and RWTU[1:0] = 1, the watchdog timer is set for 2 * 512 = 1024 system clock cycles.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>RWT[7:0]: Receive Interrupt Watchdog Timer Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This field indicates the number of system clock cycles, multiplied by factor indicated in RWTU field, for which the watchdog timer is set.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The watchdog timer is triggered with the programmed value after the Rx DMA completes the transfer of a packet for which the RI bit is not set in the ETH_DMACSR, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[30].

When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per the Interrupt Enable bit RDES3[30] of any received packet.
Channel current application transmit descriptor register  
(ETH_DMACCATXDR)

Address offset: 0x1144
Reset value: 0x0000 0000

The Channel Current Application Transmit Descriptor register points to the current Transmit descriptor read by the DMA.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **CURTDESAPTR[31:0]**: Application Transmit Descriptor Address Pointer
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Channel current application receive descriptor register  
(ETH_DMACCARXDR)

Address offset: 0x114C
Reset value: 0x0000 0000

The Channel Current Application Receive Descriptor register points to the current Receive descriptor read by the DMA.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **CURRDESAPTR[31:0]**: Application Receive Descriptor Address Pointer
The DMA updates this pointer during Rx operation. This pointer is cleared on reset.
Channel current application transmit buffer register (ETH_DMACATXBR)

Address offset: 0x1154
Reset value: 0x0000 0000

The Channel Current Application Transmit Buffer Address register points to the current Tx buffer address read by the DMA.

| Bits 31:0 CURTBUFAPTR[31:0]: Application Transmit Buffer Address Pointer |
| The DMA updates this pointer during Tx operation. This pointer is cleared on reset. |

Channel current application receive buffer register (ETH_DMACARXBR)

Address offset: 0x115C
Reset value: 0x0000 0000

The Channel Current Application Receive Buffer Address register points to the current Rx buffer address read by the DMA.

| Bits 31:0 CURRBUFFAPTR[31:0]: Application Receive Buffer Address Pointer |
| The DMA updates this pointer during Rx operation. This pointer is cleared on reset. |

Channel status register (ETH_DMACSR)

Address offset: 0x1160
Reset value: 0x0000 0000

The software driver (application) reads the Status register during interrupt service routine or polling to determine the status of the DMA.
Bits 31:22  Reserved, must be kept at reset value.

Bits 21:19  **REB[2:0]: Rx DMA Error Bits**
This field indicates the type of error that caused a Bus Error. For example, error response on the AHB interface.
- Bit [2]: Error during data transfer by Rx DMA when 1, no Error during data transfer by Rx DMA when 0.
- Bit[1]: Error during descriptor access when 1, Error during data buffer access when 0
- Bit[0]: Error during read transfer when 1, Error during write transfer when 0
This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bits 18:16  **TEB[2:0]: Tx DMA Error Bits**
This field indicates the type of error that caused a Bus Error. For example, error response on the AHB interface.
- Bit[2]: Error during data transfer by Tx DMA when 1, no Error during data transfer by Tx DMA when 0
- Bit[1]: Error during descriptor access when 1, Error during data buffer access when 0
- Bit[0]: Error during read transfer when 1, Error during write transfer when 0
This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bit 15  **NIS: Normal Interrupt Summary**
Normal Interrupt Summary bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the ETH_DMACIER register:
- Bit 0: Transmit Interrupt
- Bit 2: Transmit Buffer Unavailable
- Bit 6: Receive Interrupt
- Bit 11: Early Receive Interrupt
Only unmasked bits (interrufts for which interrupt enable is set in ETH_DMACIER register) affect the Normal Interrupt Summary bit.
This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit which causes NIS to be set is cleared.

Bit 14  **AIS: Abnormal Interrupt Summary**
Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the ETH_DMACIER register:
- Bit 1: Transmit Process Stopped
- Bit 7: Receive Buffer Unavailable
- Bit 8: Receive Process Stopped
- Bit 10: Early Transmit Interrupt
- Bit 12: Fatal Bus Error
- Bit 13: Context Descriptor Error
Only unmasked bits affect the Abnormal Interrupt Summary bit.
This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit, which causes AIS to be set, is cleared.

Bit 13  **CDE: Context Descriptor Error**
This bit indicates that the DMA Tx/Rx engine received a descriptor error, which indicates invalid context in the middle of packet flow (intermediate descriptor) or all one’s descriptor in Tx case and on Rx side it indicates DMA has read a descriptor with either of the buffer address as ones which is considered to be invalid.

Bit 12  **FBE: Fatal Bus Error**
This bit indicates that a bus error occurred (as described in the EB field). When this bit is set, the corresponding DMA channel engine disables all bus accesses.
<table>
<thead>
<tr>
<th>Bit 11</th>
<th>ERI: Early Receive Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit indicates that the DMA filled the first data buffer of the packet. The RI bit of this register automatically clears this bit.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 10</th>
<th>ETI: Early Transmit Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit indicates that the packet to be transmitted is fully transferred to the MTL Tx FIFO.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 9</th>
<th>RWT: Receive Watchdog Timeout</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled) is received.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 8</th>
<th>RPS: Receive Process Stopped</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is asserted when the Rx process enters the Stopped state.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>RBU: Receive Buffer Unavailable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit indicates that the application owns the next descriptor in the Receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the Receive Descriptor Tail Pointer register of a channel. This bit is set only when the DMA owns the previous Rx descriptor.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>RI: Receive Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit indicates that the packet reception is complete. When packet reception is complete, Bit 31 of RDES1 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. The reception remains in the Running state.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 5:3</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>TBU: Transmit Buffer Unavailable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit indicates that the application owns the next descriptor in the Transmit list, and the DMA cannot acquire it. Transmission is suspended. The TPSi field of the Debug status register (ETH_DMADSR) register explains the Transmit Process state transitions. To resume processing the Transmit descriptors, the application should do the following: 1. Change the ownership of the descriptor by setting Bit 31 of TDES3. 2. Issue a Transmit Poll Demand command. For ring mode, the application should advance the Transmit Descriptor Tail Pointer register of a channel.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>TPS: Transmit Process Stopped</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set when the transmission is stopped.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>TI: Transmit Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit indicates that the packet transmission is complete. When transmission is complete, Bit 31 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.</td>
</tr>
</tbody>
</table>
Channel missed frame count register (ETH_DMACMFCR)

Address offset: 0x116C
Reset value: 0x0000 0000

This register has the number of packet counter that got dropped by the DMA either due to Bus Error or due to programing RPF field in Channel receive control register (ETH_DMACRXCR) register.

Bits 31:16  Reserved, must be kept at reset value.

Bit 15  MFCO: Overflow status of the MFC Counter
When this bit is set then the MFC counter does not get incremented further. The bit gets cleared when this register is read.

Bits 14:11  Reserved, must be kept at reset value.

Bits 10:0  MFC[10:0]: Dropped Packet Counters
This counter indicates the number of packet counters that are dropped by the DMA either because of bus error or because of programing RPF field in Channel receive control register (ETH_DMACRXCR). The counter gets cleared when this register is read.
## Ethernet DMA register map and reset values

### Table 677. ETH_DMA common register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1000</td>
<td>ETH_DMAMR</td>
<td></td>
</tr>
<tr>
<td>0x1004</td>
<td>ETH_DMASBMR</td>
<td></td>
</tr>
<tr>
<td>0x1008</td>
<td>ETH_DMAISR</td>
<td></td>
</tr>
<tr>
<td>0x100C</td>
<td>ETH_DMADSR</td>
<td></td>
</tr>
</tbody>
</table>

### Table 678. ETH_DMA_CH register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1100</td>
<td>ETH_DMACCR</td>
<td></td>
</tr>
<tr>
<td>0x1104</td>
<td>ETH_DMACTXCR</td>
<td></td>
</tr>
<tr>
<td>0x1108</td>
<td>ETH_DMACRXR</td>
<td></td>
</tr>
<tr>
<td>0x110C-01110</td>
<td>ETH_DMACTXDLAR</td>
<td>TDESLA[31:0]</td>
</tr>
<tr>
<td>0x1114</td>
<td>ETH_DMACTXDLAR</td>
<td>RDESLA[31:0]</td>
</tr>
<tr>
<td>0x1118</td>
<td>ETH_DMACTXDTPR</td>
<td>TDT[31:0]</td>
</tr>
</tbody>
</table>

---

2830/3119  RM0481 Rev 1
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1124</td>
<td>ETH_DMACRXDTPR</td>
<td>Reserved</td>
<td>RDT[31:0]</td>
</tr>
<tr>
<td>0x1128</td>
<td>ETH_DMACCTXRLR</td>
<td>Reset value</td>
<td>TDRL[9:0]</td>
</tr>
<tr>
<td>0x112C</td>
<td>ETH_DMACRXLRL</td>
<td>Reset value</td>
<td>ARBS[7:0]</td>
</tr>
<tr>
<td>0x1130</td>
<td>ETH_DMACIER</td>
<td>Reset value</td>
<td>RDRL[9:0]</td>
</tr>
<tr>
<td>0x1134</td>
<td>ETH_DMACIWER</td>
<td>Reset value</td>
<td>RWTU[1:0]</td>
</tr>
<tr>
<td>0x1138</td>
<td>ETH_DMACRIWTR</td>
<td>Reset value</td>
<td>RWT[7:0]</td>
</tr>
<tr>
<td>0x113C-</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1140</td>
<td>ETH_DMACCATXDR</td>
<td></td>
<td>CURTDESAPTR[31:0]</td>
</tr>
<tr>
<td>0x1148</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x114C</td>
<td>ETH_DMACATXDR</td>
<td>Reset value</td>
<td>CURRDESAPTR[31:0]</td>
</tr>
<tr>
<td>0x1150</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1154</td>
<td>ETH_DMACATXBR</td>
<td>Reset value</td>
<td>CURTBUFAPTR[31:0]</td>
</tr>
<tr>
<td>0x1158</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x115C</td>
<td>ETH_DMACATXBR</td>
<td>Reset value</td>
<td>CURRBUFAPTR[31:0]</td>
</tr>
<tr>
<td>0x1160</td>
<td>ETH_DMACSR</td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x1164-</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Refer to *Section 2.3 on page 111* for the register boundary addresses.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x116C</td>
<td>ETH_DMACMFCR</td>
<td></td>
<td>31</td>
<td>0x00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MFCO</td>
<td></td>
<td>30</td>
<td>0x00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MFC[10:0]</td>
<td></td>
<td>29</td>
<td>0x00000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>28</td>
<td>0x00000000</td>
<td></td>
</tr>
</tbody>
</table>
57.11.3 Ethernet MTL registers

Operating mode Register (ETH_MTLOMR)

Address offset: 0x0C00
Reset value: 0x0000 0000

The Operating Mode register establishes the Transmit and Receive operating modes and commands.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CNTCLR**: Counters Reset
When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. If this bit is set along with CNTPRST bit, CNTPRST has precedence.

Bit 8 **CNTPRST**: Counters Preset
When this bit is set:
- Tx queue underflow register (ETH_MTLTXQUR) is initialized/preset to 0x7F0.
- Missed Packet and Overflow Packet counters in Rx queue missed packet and overflow counter register (ETH_MTLRXQMPOCR) is initialized/preset to 0x7F0

This bit is cleared automatically.

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 Reserved, must be kept at reset value.

Bit 1 **DTXSTS**: Drop Transmit Status
When this bit is set, the Tx packet status received from the MAC is dropped in the MTL. When this bit is reset, the Tx packet status received from the MAC is forwarded to the application.

Bit 0 Reserved, must be kept at reset value.
Interrupt status Register (ETH_MTLISR)

Address offset: 0x0C20

Reset value: 0x0000 0000

The software driver (application) reads this register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **Q0IS**: Queue interrupt status

This bit indicates that an interrupt has been generated by Queue. To reset this bit, read ETH_MTLQICSR register to identify the interrupt cause and clear the source.

Tx queue operating mode Register (ETH_MTLTXQOMR)

Address offset: 0x0D00

Reset value: 0x0007 0008

The Queue Transmit Operating Mode register establishes the Transmit queue operating modes and commands.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **TQS[2:0]**: Transmit queue size

This field indicates the size of the allocated transmit queues in blocks of 256 bytes.

Queue size range from 256 bytes (TQS=0b000) to 2048 bytes (TQS=0b111).

Bits 15:7 Reserved, must be kept at reset value.
Bits 6:4 **TTC[2:0]**: Transmit Threshold Control
These bits control the threshold level of the MTL Tx queue. The transmission starts when the packet size within the MTL Tx queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset.

- 000: 32
- 001: 64
- 010: 96
- 011: 128
- 100: 192
- 101: 256
- 110: 384
- 111: 512

Bits 3:2 **TXQEN[1:0]**: Transmit Queue Enable
This field is used to enable/disable the transmit queue.

- 00: Not enabled
- 10: Enabled

Others: Reserved, must not be used.

**Note**: In multiple Tx queues configuration, all the queues are disabled by default. Enable the Tx queue by programming this field.

Bit 1 **TSF**: Transmit Store and Forward
When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[6:4] of this register are ignored. This bit should be changed only when the transmission is stopped.

Bit 0 **FTQ**: Flush Transmit Queue
When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the ETH_MTLTXQOMR register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission.

**Note**: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (eth_mii_tx_clk) should be active.

**Tx queue underflow register (ETH_MTLTXQUR)**
Address offset: 0x0D04
Reset value: 0x0000 0000

The Queue Underflow Counter register contains the counter for packets aborted because of Transmit queue underflow and packets missed because of Receive queue packet flush.
**Tx queue debug register (ETH_MTLTXQDR)**

Address offset: 0x0D08  
Reset value: 0x0000 0000

The Queue Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

<table>
<thead>
<tr>
<th>Bits 31:23</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bits 22:20 | **STXSTSF[2:0]: Number of Status Words in Tx Status FIFO of Queue**  
This field indicates the current number of status in the Tx Status FIFO of this queue.  
When the DTXSTS bit of ETH_MTLOMR register is set to 1, this field does not reflect the number of status words in Tx Status FIFO. |
| Bit 19     | Reserved, must be kept at reset value. |
| Bits 18:16 | **PTXQ[2:0]: Number of Packets in the Transmit Queue**  
This field indicates the current number of packets in the Tx queue.  
When the DTXSTS bit of **Operating mode Register (ETH_MTLOMR)** register is set to 1, this field does not reflect the number of packets in the Transmit queue. |
| Bits 15:6  | Reserved, must be kept at reset value. |
| Bit 5      | **TXSTSFSTS**: MTL Tx Status FIFO Full Status  
When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission. |
| Bit 4      | **TXQSTS**: MTL Tx Queue Not Empty Status  
When this bit is high, it indicates that the MTL Tx queue is not empty and some data is left for transmission. |
Bit 3 **TWCSTS**: MTL Tx Queue Write Controller Status
When high, this bit indicates that the MTL Tx queue Write Controller is active, and it is transferring the data to the Tx queue.

Bits 2:1 **TRCSTS[1:0]**: MTL Tx Queue Read Controller Status
This field indicates the state of the Tx Queue Read Controller:
00: Idle state
01: Read state (transferring data to the MAC transmitter)
10: Waiting for pending Tx Status from the MAC transmitter
11: Flushing the Tx queue because of the Packet Abort request from the MAC

Bit 0 **TXQPAUSED**: Transmit Queue in Pause
When this bit is high and the Rx flow control is enabled, it indicates that the Tx queue is in the Pause condition (in the Full-duplex only mode) because of the following:
- Reception of the PFC packet for the priorities assigned to the Tx queue when PFC is enabled
- Reception of 802.3x Pause packet when PFC is disabled

**Queue interrupt control status Register (ETH_MTLQICSR)**

Address offset: 0x0D2C
Reset value: 0x0000 0000

This register contains the interrupt enable and status bits for the queue interrupts.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RXOIE: Receive Queue Overflow Interrupt Enable</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>RXOVFS: Receive Queue Overflow Interrupt Status</td>
<td>rc_w1</td>
</tr>
<tr>
<td>29</td>
<td>TXUIE: Transmit Queue in Pause</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>TXUNFS: Transmit Queue in Pause</td>
<td>rc_w1</td>
</tr>
<tr>
<td>27</td>
<td>Bit 24</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Bit 23</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Bit 22</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Bit 21</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Bit 20</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Bit 19</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Bit 18</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Bit 17</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Bit 16</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Bit 15</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Bit 14</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Bit 13</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Bit 12</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Bit 11</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Bit 10</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Bit 9</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Bit 8</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Bit 7</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Bit 6</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Bit 5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Bit 4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Bit 3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Bit 2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Bit 1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Bit 0</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RXOIE**: Receive Queue Overflow Interrupt Enable
When this bit is set, the Receive Queue Overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **RXOVFS**: Receive Queue Overflow Interrupt Status
This bit indicates that the Receive Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit.

Bits 15:9 Reserved, must be kept at reset value.
Bit 8 **TXUIE**: Transmit Queue Underflow Interrupt Enable
When this bit is set, the Transmit Queue Underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **TXUNFIS**: Transmit Queue Underflow Interrupt Status
This bit indicates that the Transmit Queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit.
Rx queue operating mode register (ETH_MTLRXQOMR)

Address offset: 0x0D30
Reset value: 0x0070 0000

The Queue Receive operating Mode register establishes the Receive queue operating modes and command.

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQS[2:0]</td>
<td>22:20</td>
<td>Receive Queue Size</td>
</tr>
<tr>
<td>DIS_TCP_EF</td>
<td>7</td>
<td>Disable Dropping of TCP/IP Checksum Error Packets</td>
</tr>
<tr>
<td>RSF</td>
<td>6</td>
<td>Receive Queue Store and Forward</td>
</tr>
<tr>
<td>FEP</td>
<td>5</td>
<td>Forward Error Packets</td>
</tr>
<tr>
<td>FUP</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>RTC[1:0]</td>
<td>3:0</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **RQS[2:0]**: Receive Queue Size

This field is read-only and the configured Rx FIFO size in blocks of 256 bytes is reflected in the reset value. The size of the Queue is (RQS + 1) * 256 bytes.

Bits 19:7 Reserved, must be kept at reset value.

Bit 6 **DIS_TCP_EF**: Disable Dropping of TCP/IP Checksum Error Packets

When this bit is set, the MAC does not drop the packets which only have the errors detected by the Receive Checksum Offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC.

When this bit is reset, all error packets are dropped if the FEP bit is reset.

Bit 5 **RSF**: Receive Queue Store and Forward

When this bit is set, the Ethernet peripheral reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register.

Bit 4 **FEP**: Forward Error Packets

When this bit is reset, the Rx queue drops packets with error status (CRC error, receive error, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped.

When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA.
Bit 3 **FUP**: Forward Undersized Good Packets
When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **RTC[1:0]**: Receive Queue Threshold Control
These bits control the threshold level of the MTL Rx queue (in bytes):

- 00: 64
- 01: 32
- 10: 96
- 11: 128

The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred.

This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1.
Rx queue missed packet and overflow counter register
(ETH_MTLRXQMPOCR)

Address offset: 0x0D34
Reset value: 0x0000 0000

The Queue missed packet and overflow counter registers contain the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>-----------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>-----------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MISCNTOVF**: Missed Packet Counter Overflow Bit
When set, this bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit.

Bits 26:16 **MISPKTCNT[10:0]**: Missed Packet Counter
This field indicates the number of packets missed by the Ethernet peripheral because the application requested to flush the packets for this queue. This counter is reset when this register is read.
This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **OVFCNTOVF**: Overflow Counter Overflow Bit
When set, this bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit.

Bits 10:0 **OVFPKTCNT[10:0]**: Overflow Packet Counter
This field indicates the number of packets discarded by the Ethernet peripheral because of Receive queue overflow. This counter is incremented each time the Ethernet peripheral discards an incoming packet because of overflow. This counter is reset when this register is read.
Rx queue debug register (ETH_MTLRXQDR)

Address offset: 0x0D38
Reset value: 0x0000 0000

The Queue Receive Debug register gives the debug status of various blocks related to the Receive queue.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 15</td>
<td>Bit 14</td>
<td>Bit 13</td>
<td>Bit 12</td>
<td>Bit 11</td>
<td>Bit 10</td>
<td>Bit 9</td>
<td>Bit 8</td>
<td>Bit 7</td>
<td>Bit 6</td>
<td>Bit 5</td>
<td>Bit 4</td>
<td>Bit 3</td>
<td>Bit 2</td>
<td>Bit 1</td>
<td>Bit 0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:30  Reserved, must be kept at reset value.

Bits 29:16  **PRXQ[13:0]: Number of Packets in Receive Queue**
This field indicates the current number of packets in the Rx queue. The theoretical maximum value for this field is 256Kbyte/16bytes = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.

Bits 15:6  Reserved, must be kept at reset value.

Bits 5:4  **RXQSTS[1:0]: MTL Rx Queue Fill-Level Status**
This field gives the status of the fill-level of the Rx queue:
00: Rx queue empty
01: Rx queue fill-level below flow-control deactivate threshold
10: Rx queue fill-level above flow-control activate threshold
11: Rx queue full

Bit 3  Reserved, must be kept at reset value.

Bits 2:1  **RRCSTS[1:0]: MTL Rx Queue Read Controller State**
This field gives the state of the Rx queue Read controller:
00: Idle state
01: Reading packet data
10: Reading packet status (or timestamp)
11: Flushing the packet data and status

Bit 0  **RWCSTS: MTL Rx Queue Write Controller Active Status**
When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx queue.
## Ethernet MTL register map and reset values

Table 679. ETH_MTL register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0C00</td>
<td>ETH_MTLOMR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td>0x0C04-0x0C1C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C20</td>
<td>ETH_MTLISR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x0C24-0x0CFC</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C40</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0D00</td>
<td>ETH_MTLTXQOMR</td>
<td>1 1 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 1 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0x0D04</td>
<td>ETH_MTLTXQUR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0x0D08</td>
<td>ETH_MTLTXQDR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0x0DCC-0x0D28</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0D2C</td>
<td>ETH_MTLQICSR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x0D30</td>
<td>ETH_MTLRXQOMR</td>
<td>1 1 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

STI
Table 679. ETH_MTL register map and reset values (continued)

| Offset   | Register name   | reset value | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|-----------------|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xD34    | ETH_MTLRXQMP0CR |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | MISCNTVF        | 0           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|          | MISPKTCNT[10:0] | 0           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|          | OVFCTRVF        | 0           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|          | OVFPKTCNT[10:0] | 0           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|          | Reset value      | 0           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xD38    | ETH_MTLRXQDR    |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | PRXQ[13:0]      | 0           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|          | RISSOF          | 0           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|          | RISCST[1:0]     | 0           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|          | RISCSO          | 0           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|          | Reset value      | 0           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xD3C-0xD58 | Reserved       |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.3 on page 111 for the register boundary addresses.
### 57.11.4 Ethernet MAC and MMC registers

**Operating mode configuration register (ETH_MACCR)**

Address offset: 0x0000

Reset value: 0x0000 0000

The MAC Configuration Register establishes the operating mode of the MAC.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>ARPEN</td>
<td>ARP Offload Enable</td>
</tr>
<tr>
<td>30-28</td>
<td>SARC[2:0]</td>
<td>Source Address Insertion or Replacement Control</td>
</tr>
<tr>
<td>27-25</td>
<td>IPC,H[7:0]</td>
<td>Paket Length Control</td>
</tr>
<tr>
<td>24-22</td>
<td>IPG[15:0]</td>
<td>Inter Packet Gap</td>
</tr>
<tr>
<td>21-19</td>
<td>GPSLCE[2:0]</td>
<td>Packet Length Control</td>
</tr>
<tr>
<td>18-16</td>
<td>SKP</td>
<td>Collision Count Control</td>
</tr>
<tr>
<td>15-13</td>
<td>CST</td>
<td>Control Shift Register</td>
</tr>
<tr>
<td>12-10</td>
<td>ACS</td>
<td>Accept Count Control</td>
</tr>
<tr>
<td>9-7</td>
<td>WD</td>
<td>Wait Count Control</td>
</tr>
<tr>
<td>6-4</td>
<td>Res.</td>
<td>Reserved</td>
</tr>
<tr>
<td>3-1</td>
<td>JD</td>
<td>junction delay</td>
</tr>
<tr>
<td>0</td>
<td>JE</td>
<td>Error mode</td>
</tr>
</tbody>
</table>

**Bit 31 ARPEN:** ARP Offload Enable

When this bit is set, the MAC can recognize an incoming ARP request packet and schedules the ARP packet for transmission. It forwards the ARP packet to the application and also indicate the events in the RxStatus.

When this bit is reset, the MAC receiver does not recognize any ARP packet and indicates them as Type frame in the RxStatus.

**Bits 30:28 SARC[2:0]:** Source Address Insertion or Replacement Control

This field controls the source address insertion or replacement for all transmitted packets. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits[29:28]:

- 010: the MAC inserts the content of the MAC Address 0 registers (`MAC Address 0 high register (ETH_MACA0HR)` and `MAC Address x low register (ETH_MACAxLR)`) in the SA field of all transmitted packets.
- 011: the MAC replaces the content of the MAC Address 0 registers (`MAC Address 0 high register (ETH_MACA0HR)` and `MAC Address x low register (ETH_MACAxLR)`) in the SA field of all transmitted packets.
- 110: the MAC inserts the content of the MAC Address 1 registers (`MAC Address x high register (ETH_MACAxHR)` and `MAC Address x low register (ETH_MACAxLR)`) in the SA field of all transmitted packets.
- 111: the MAC replaces the content of the MAC Address 1 registers (`MAC Address x high register (ETH_MACAxHR)` and `MAC Address x low register (ETH_MACAxLR)`) in the SA field of all transmitted packets.
- Others: Reserved, must not be used.

**Note:** Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.
Bit 27  **IPC**: Checksum Offload

When set, this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled.

The Layer 3 and Layer 4 Packet Filter feature automatically selects the IPC Full Checksum Offload Engine on the Receive side. When this feature is enabled, you must set the IPC bit.

Bits 26:24  **IPG[2:0]**: Inter-Packet Gap

These bits control the minimum IPG between packets during transmission.

- 000: 96 bit times
- 001: 88 bit times
- 010: 80 bit times
- ...
- 111: 40 bit times

This range of minimum IPG is valid in Full-duplex mode. In the Half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered.

When a JAM pattern is being transmitted because of backpressure activation, the MAC does not consider the minimum IPG.

The above function (IPG less than 96 bit times) is valid only when EIPGEN bit in ETH_MACECR register is reset. When EIPGEN is set, then the minimum IPG (greater than 96 bit times) is controlled as per the description given in EIPG field in ETH_MACECR register.

Bit 23  **GPSLCE**: Giant Packet Size Limit Control Enable

When this bit is set, the MAC considers the value in GPSL field in ETH_MACECR register to declare a received packet as Giant packet. This field must be programmed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit.

When this bit is reset, the MAC considers a received packet as Giant packet when its size is greater than 1,518 bytes (1522 bytes for tagged packet).

The watchdog timeout limit, Jumbo Packet Enable and 2K Packet Enable have higher precedence over this bit, that is the MAC considers a received packet as Giant packet when its size is greater than 9,018 bytes (9,022 bytes for tagged packet) with Jumbo Packet Enabled and greater than 2,000 bytes with 2K Packet Enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programmed giant packet limit should be less than the watchdog limit to get the giant packet status.

Bit 22  **S2KP**: IEEE 802.3as Support for 2K Packets

When this bit is set, the MAC considers all packets with up to 2,000 bytes length as normal packets. When the JE bit is not set, the MAC considers all received packets of size more than 2K bytes as Giant packets.

When this bit is reset and the JE bit is not set, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. For more information about how the setting of this bit and the JE bit impact the Giant packet status, see Table 680: Giant Packet Status based on S2KP and JE Bits.

**Note**: When the JE bit is set, setting this bit has no effect on the giant packet status.

Bit 21  **CST**: CRC stripping for Type packets

When this bit is set, the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver.

This function is valid when Type 2 Checksum Offload Engine is enabled.

**Note**: For information about how the settings of the ACS bit and this bit impact the packet length, see Table 681: Packet Length based on the CST and ACS bits.
Bit 20  **ACS**: Automatic Pad or CRC Stripping
When this bit is set, the MAC strips the Pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field.
When this bit is reset, the MAC passes all incoming packets to the application, without any modification.

*Note:* For information about how the settings of CST bit and this bit impact the packet length, see Table 681: Packet Length based on the CST and ACS bits.

Bit 19  **WD**: Watchdog Disable
When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive packets of up to 16,383 bytes.
When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the packet being received. The MAC cuts off any bytes received after 2,048 bytes.

Bit 18  Reserved, must be kept at reset value.

Bit 17  **JD**: Jabber Disable
When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer packets of up to 16,383 bytes.
When this bit is reset, if the application sends more than 2,048 bytes of data (10,240 if JE is set high) during transmission, the MAC does not send rest of the bytes in that packet.

Bit 16  **JE**: Jumbo Packet Enable
When this bit is set, the MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN tagged packets) without reporting a giant packet error in the Rx packet status.
For more information about how the setting of this bit and the JE bit impact the Giant packet status, see Table 680: Giant Packet Status based on S2KP and JE Bits.

Bit 15  Reserved, must be kept at reset value.

Bit 14  **FES**: MAC Speed
This bit selects the speed in the 10/100 Mbps mode:
0: 10 Mbps
1: 100 Mbps

Bit 13  **DM**: Duplex Mode
When this bit is set, the MAC operates in the Full-duplex mode in which it can transmit and receive simultaneously.

Bit 12  **LM**: Loopback Mode
When this bit is set, the MAC operates in the loopback mode at MII. The MII Rx clock input (eth_mii_rx_clk) is required for the loopback to work properly. This is because the Tx clock is not internally looped back.

Bit 11  **ECRSFD**: Enable Carrier Sense Before Transmission in Full-duplex mode
When this bit is set, the MAC transmitter checks the CRS signal before packet transmission in the Full-duplex mode. The MAC starts the transmission only when the CRS signal is low.
When this bit is reset, the MAC transmitter ignores the status of the CRS signal.

Bit 10  **DO**: Disable Receive Own
When this bit is set, the MAC disables the reception of packets when the ETH_TX_EN is asserted in the Half-duplex mode. When this bit is reset, the MAC receives all packets given by the PHY.
This bit is not applicable in the Full-duplex mode. This bit is reserved and read-only (RO) with default value in the Full-duplex-only configurations.
Bit 9 **DCRS**: Disable Carrier Sense During Transmission
When this bit is set, the MAC transmitter ignores the MII CRS signal during packet transmission in the Half-duplex mode. As a result, no errors are generated because of Loss of Carrier or No Carrier during transmission.
When this bit is reset, the MAC transmitter generates errors because of Carrier Sense. The MAC can even abort the transmission.

Bit 8 **DR**: Disable Retry
When this bit is set, the MAC attempts only one transmission. When a collision occurs on the MII interface, the MAC ignores the current packet transmission and reports a Packet Abort with excessive collision error in the Tx packet status.
When this bit is reset, the MAC retries based on the settings of the BL field. This bit is applicable only in the Half-duplex mode.

Bit 7 Reserved, must be kept at reset value.

Bits 6:5 **BL[1:0]**: Back-Off Limit
The back-off limit determines the random integer number (r) of slot time delays (512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision:
- **00**: \( k = \min (n, 10) \)
- **01**: \( k = \min (n, 8) \)
- **10**: \( k = \min (n, 4) \)
- **11**: \( k = \min (n, 1) \)
where \( n \) = retransmission attempt
The random integer \( r \) takes the value in the range \( 0 \leq r < 2^k \).
This bit is applicable only in the Half-duplex mode.

Bit 4 **DC**: Deferral Check
When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Packet Abort status, along with the excessive deferral error bit set in the Tx packet status, when the Tx state machine is deferred for more than 24,288 bit times in 10 or 100 Mbps mode.
Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on MII.
The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0, and it is restarted.
When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive.
This bit is applicable only in the Half-duplex mode.

Bits 3:2 **PRELEN[1:0]**: Preamble Length for Transmit packets
These bits control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the Full-duplex mode.
- **00**: 7 bytes of preamble
- **01**: 5 bytes of preamble
- **10**: 3 bytes of preamble
- **11**: Reserved, must not be used
Bit 1 **TE**: Transmitter Enable
When this bit is set, the Tx state machine of the MAC is enabled for transmission on the MII interface. When this bit is reset, the MAC Tx state machine is disabled after it completes the transmission of the current packet. The Tx state machine does not transmit any more packets.

Bit 0 **RE**: Receiver Enable
When this bit is set, the Rx state machine of the MAC is enabled for receiving packets from the MII interface. When this bit is reset, the MAC Rx state machine is disabled after it completes the reception of the current packet. The Rx state machine does not receive any more packets from the MII interface.

*Table 680* shows how the settings of S2KP and JE bits of the ETH_MACCR register impact the giant packet status.

*Table 680. Giant Packet Status based on S2KP and JE Bits*(1)

<table>
<thead>
<tr>
<th>Length/Type Field</th>
<th>Received Packet Length</th>
<th>S2KP</th>
<th>JE</th>
<th>Giant Packet Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Untagged packet</td>
<td>&gt; 1,518</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>&gt; 2,000</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>&gt; 9,018</td>
<td>x</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>VLAN tagged packet</td>
<td>&gt; 1,522</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>&gt; 2,000</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>&gt; 9,022</td>
<td>x</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. For all other combinations, the Giant Packet status is 0.

*Table 681* shows how the settings of the CST and ACS bits of the ETH_MACCR register impact whether CRC length is included in the packet length.

*Table 681. Packet Length based on the CST and ACS bits*

<table>
<thead>
<tr>
<th>Received Packet Length</th>
<th>CST</th>
<th>ACS</th>
<th>FCS Stripping Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 1,536</td>
<td>x</td>
<td>0</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>1</td>
<td>Yes (for Ethernet packets)</td>
</tr>
<tr>
<td>≥ 1,536</td>
<td>0</td>
<td>x</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>x</td>
<td>Yes (for Type packets)</td>
</tr>
</tbody>
</table>
Extended operating mode configuration register (ETH_MACECR)

Address offset: 0x0004
Reset value: 0x0000 0000

The MAC Extended Configuration Register establishes the operating mode of the MAC.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:30: Reserved, must be kept at reset value.

Bits 29:25: **EIPG[4:0]:** Extended Inter-Packet Gap

The value in this field is applicable when the EIPGEN bit is set. This field (as Most Significant bits) along with IPG field in Operating mode configuration register (ETH_MACCR), gives the minimum IPG greater than 96 bit times in steps of 8 bit times. For example:

- EIPG = 0 and IPG = 0 give 104 bit times
- EIPG = 0 and IPG = 1 give 112 bit times
- EIPG = 0 and IPG = 2 give 120 bit times
- ..
- EIPG = 7 and IPG = 31 give 2144 bit times

Bit 24: **EIPGEN:** Extended Inter-Packet Gap Enable

When this bit is set, the MAC interprets EIPG field and IPG field in Operating mode configuration register (ETH_MACCR) as minimum IPG greater than 96 bit times in steps of 8 bit times.

When this bit is reset, the MAC ignores EIPG field and interprets IPG field in Operating mode configuration register (ETH_MACCR) as minimum IPG less than or equal to 96 bit times in steps of 8 bit times.

*Note:* The extended Inter-Packet Gap feature must be enabled when operating in Full-duplex mode only. There may be undesirable effects on back-pressure function and frame transmission if it is enabled in Half-duplex mode.

Bits 23:19: Reserved, must be kept at reset value.

Bit 18: **USP:** Unicast Slow Protocol Packet Detect

When this bit is set, the MAC detects the Slow Protocol packets with unicast address of the station specified in the MAC Address 0 high register (ETH_MACA0HR) and MAC Address 0 low register MAC Address x low register (ETH_MACAxLR). The MAC also detects the Slow Protocol packets with the Slow Protocols multicast address (01-80-C2-00-00-02).

When this bit is reset, the MAC detects only Slow Protocol packets with the Slow Protocol multicast address specified in the IEEE 802.3-2008, Section 5.
Bit 17  **SPEN**: Slow Protocol Detection Enable
When this bit is set, MAC processes the Slow Protocol packets (Ether Type 0x8809) and provides the Rx status. The MAC discards the Slow Protocol packets with invalid subtypes. When this bit is reset, the MAC forwards all error-free Slow Protocol packets to the application. The MAC considers such packets as normal Type packets.

Bit 16  **DCRCC**: Disable CRC Checking for Received Packets
When this bit is set, the MAC receiver does not check the CRC field in the received packets. When this bit is reset, the MAC receiver always checks the CRC field in the received packets.

Bits 15:14  Reserved, must be kept at reset value.

Bits 13:0  **GPSL[13:0]**: Giant Packet Size Limit
If the received packet size is greater than the value programmed in this field in units of bytes, the MAC declares the received packet as Giant packet. The value programmed in this field must be greater than or equal to 1,518 bytes. Any other programmed value is considered as 1,518 bytes.
For VLAN tagged packets, the MAC adds 4 bytes to the programmed value. For double VLAN tagged packets, the MAC adds 8 bytes to the programmed value. The value in this field is applicable when the GPSLCE bit is set in ETH_MACCR register.

**Packet filtering control register (ETH_MACPFR)**
Address offset: 0x0008
Reset value: 0x0000 0000

The MAC Packet Filter register contains the filter controls for receiving packets. Some of the controls from this register go to the address check block of the MAC which performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as Pass Bad Packets and Pass Control Packets.

Bit 31  **RA**: Receive All
When this bit is set, the MAC Receiver module passes all received packets to the application, irrespective of whether they pass the address filter or not. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bit in the Rx Status Word.
When this bit is reset, the Receiver module passes only those packets to the application that pass the SA or DA address filter.

Bits 30:22  Reserved, must be kept at reset value.

Bit 21  **DNTU**: Drop Non-TCP/UDP over IP Packets
When this bit is set, the MAC drops the non-TCP or UDP over IP packets. The MAC forward only those packets that are processed by the Layer 4 filter. When this bit is reset, the MAC forwards all non-TCP or UDP over IP packets.
Bit 20 **IPFE**: Layer 3 and Layer 4 Filter Enable
When this bit is set, the MAC drops packets that do not match the enabled Layer 3 and Layer 4 filters. If Layer 3 or Layer 4 filters are not enabled for matching, this bit does not have any effect.
When this bit is reset, the MAC forwards all packets irrespective of the match status of the Layer 3 and Layer 4 fields.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **VTFE**: VLAN Tag Filter Enable
When this bit is set, the MAC drops the VLAN tagged packets that do not match the VLAN Tag. When this bit is reset, the MAC forwards all packets irrespective of the match status of the VLAN Tag.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **HPF**: Hash or Perfect Filter
When this bit is set, the address filter passes a packet if it matches either the perfect filtering or Hash filtering as set by the HMC or HUC bit.
When this bit is reset and the HUC or HMC bit is set, the packet is passed only if it matches the Hash filter.

Bit 9 **SAF**: Source Address Filter Enable
When this bit is set, the MAC compares the SA field of the received packets with the values programmed in the enabled SA registers. If the comparison fails, the MAC drops the packet.
When this bit is reset, the MAC forwards the received packet to the application with updated SAF bit of the Rx Status depending on the SA address comparison.

*Note*: According to the IEEE specification, Bit 47 of the SA is reserved. However, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.

Bit 8 **SAIF**: SA Inverse Filtering
When this bit is set, the Address Check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programmed in the SA registers, it is marked as failing the SA Address filter.
When this bit is reset, if the SA of a packet does not match the values programmed in the SA registers, it is marked as failing the SA Address filter.

Bits 7:6 **PCF[1:0]**: Pass Control Packets
These bits control the forwarding of all control packets (including unicast and multicast Pause packets).
00: The MAC filters all control packets from reaching the application.
01: The MAC forwards all control packets except Pause packets to the application even if they fail the Address filter.
10: The MAC forwards all control packets to the application even if they fail the Address filter.
11: The MAC forwards the control packets that pass the Address filter.

Bit 5 **DBF**: Disable Broadcast Packets
When this bit is set, the AFM module blocks all incoming broadcast packets. In addition, it overrides all other filter settings.
When this bit is reset, the AFM module passes all received broadcast packets.

Bit 4 **PM**: Pass All Multicast
When this bit is set, it indicates that all received packets with a multicast destination address (first bit in the destination address field is ‘1’) are passed. When this bit is reset, filtering of multicast packet depends on HMC bit.
Bit 3 **DAIF**: DA Inverse Filtering
When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast packets. When this bit is reset, normal filtering of packets is performed.

Bit 2 **HMC**: Hash Multicast
When this bit is set, the MAC performs the destination address filtering of received multicast packets according to the Hash table.
When this bit is reset, the MAC performs the perfect destination address filtering for multicast packets, that is, it compares the DA field with the values programmed in DA registers.

Bit 1 **HUC**: Hash Unicast
When this bit is set, the MAC performs the destination address filtering of unicast packets according to the Hash table.
When this bit is reset, the MAC performs a perfect destination address filtering for unicast packets, that is, it compares the DA field with the values programmed in DA registers.

Bit 0 **PR**: Promiscuous Mode
When this bit is set, the Address Filtering module passes all incoming packets irrespective of the destination or source address. The SA or DA Filter Fails status bits of the Rx Status Word are always cleared when PR is set.

**Watchdog timeout register (ETH_MACWTR)**
Address offset: 0x000C
Reset value: 0x0000 0000

The Watchdog Timeout register controls the watchdog timeout for received packets.
Hash Table 0 register (ETH_MACHT0R)

Address offset: 0x0010
Reset value: 0x0000 0000

The Hash Table Register 0 contains the first lower 32 bits of the Hash table (64 bits).

The Hash table is used for group address filtering.

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1) and the least significant five bits determine the bit within the register. For example, a hash value of 0b100000 selects Bit 0 of the Hash Table Register 1.

The Hash value of the destination address is calculated in the following way:
1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

Bits 31:9  Reserved, must be kept at reset value.

Bit 8  PWE: Programmable Watchdog Enable
When this bit is set and the WD bit of the Operating mode configuration register (ETH_MACCR) register is reset, the WTO field is used as watchdog timeout for a received packet. When this bit is cleared, the watchdog timeout for a received packet is controlled by setting of WD and JE bits in Operating mode configuration register (ETH_MACCR) register.

Bits 7:4  Reserved, must be kept at reset value.

Bits 3:0  WTO[3:0]: Watchdog Timeout
When the PWE bit is set and the WD bit of the Operating mode configuration register (ETH_MACCR) register is reset, this field is used as watchdog timeout for a received packet.

If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet.

Encoding is as follows:
0x0: 2 Kbytes
0x1: 3 Kbytes
0x2: 4 Kbytes
0x3: 5 Kbytes
...
0xC: 14 Kbytes
0xD: 15 Kbytes
0xE: 16383 Bytes
0xF: Reserved, must not be used

Note: When the PWE bit is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped.
Hash Table 1 register (ETH_MACHT1R)

Address offset: 0x0014
Reset value: 0x0000 0000

The Hash Table 1 register contains the upper 32 bits of the Hash table (64 bits).

The Hash table is used for group address filtering.

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1) and the least significant five bits determine the bit within the register.

For example, a hash value of 6'b100000 selects Bit 0 of the Hash Table Register 1.

The Hash value of the destination address is calculated in the following way:
1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

Bits 31:0 HT31T0[31:0]: MAC Hash Table First 32 Bits
This field contains the first 32 Bits [31:0] of the Hash table.

Bits 31:0 HT63T32[31:0]: MAC Hash Table Second 32 Bits
This field contains the second 32 Bits [63:32] of the Hash table.
VLAN tag register (ETH_MACVTR)

Address offset: 0x0050
Reset value: 0x0000 0000

The VLAN Tag register identifies the IEEE 802.1Q VLAN type packets.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>EIVLRXS</th>
<th>Enable Inner VLAN Tag in Rx Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>When this bit is set, the MAC provides the inner VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the inner VLAN Tag in Rx status.</td>
</tr>
</tbody>
</table>

| Bit 30 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bits 29:28</th>
<th>EIVLS[1:0]</th>
<th>Enable Inner VLAN Tag Stripping on Receive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00: Do not strip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>01: Strip if VLAN filter passes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10: Strip if VLAN filter fails</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11: Always strip</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 27</th>
<th>ERIVLT</th>
<th>Enable Inner VLAN Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag (if present). When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag (if present). The ERSVLM bit determines which VLAN type is enabled for filtering or matching. The ERSVLM bit and DOVLTC bit determines which VLAN type is enabled for filtering.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 26</th>
<th>EDVLP</th>
<th>Enable Double VLAN Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When this bit is set, the MAC enables processing of up to two VLAN Tags on Tx and Rx (if present). When this bit is reset, the MAC enables processing of up to one VLAN Tag on Tx and Rx (if present).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 25</th>
<th>VTHM</th>
<th>VLAN Tag Hash Table Match Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When this bit is set, the most significant four bits of CRC of VLAN Tag are used to index the content of the ETH_MACVLANHTR register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the packet matched the VLAN Hash table. When the ETV bit is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison. When the ETV bit is reset, the CRC of the 16-bit VLAN tag is used for comparison. When this bit is reset, the VLAN Hash Match operation is not performed.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 24</th>
<th>EVLRXS</th>
<th>Enable VLAN Tag in Rx status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When this bit is set, MAC provides the outer VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the outer VLAN Tag in Rx status.</td>
<td></td>
</tr>
</tbody>
</table>

| Bit 23 | Reserved, must be kept at reset value. |
Bits 22:21  **EVLS[1:0]: Enable VLAN Tag Stripping on Receive**
- This field indicates the stripping operation on the outer VLAN Tag in received packet:
  - 00: Do not strip
  - 01: Strip if VLAN filter passes
  - 10: Strip if VLAN filter fails
  - 11: Always strip

Bit 20  **DOVLTC: Disable VLAN Type Check**
- When this bit is set, the MAC does not check whether the VLAN Tag specified by the ERIVLT bit is of type S-VLAN or C-VLAN.
- When this bit is reset, the MAC filters or matches the VLAN Tag specified by the ERIVLT bit only when VLAN Tag type is similar to the one specified by the ERSVLM bit.

Bit 19  **ERSVLM: Enable Receive S-VLAN Match**
- When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets.
- The ERIVLT bit determines the VLAN tag position considered for filtering or matching.

Bit 18  **ESVL: Enable S-VLAN**
- When this bit is set, the MAC transmitter and receiver consider the S-VLAN packets (Type = 0x88A8) as valid VLAN tagged packets.

Bit 17  **VTIM: VLAN Tag Inverse Match Enable**
- When this bit is set, this bit enables the VLAN Tag inverse matching. The packets without matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The packets with matched VLAN Tag are marked as matched.

Bit 16  **ETV: Enable 12-Bit VLAN Tag Comparison**
- When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits[11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet. Similarly, when enabled, only 12 bits of the VLAN tag in the received packet are used for Hash-based VLAN filtering.
- When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for comparison and VLAN Hash filtering.

Bits 15:0  **VL[15:0]: VLAN Tag Identifier for Receive Packets**
- This field contains the 802.1Q VLAN tag to identify the VLAN packets. This VLAN tag identifier is compared to the 15th and 16th bytes of the packets being received for VLAN packets. The following list describes the bits of this field:
  - Bits[15:13]: User Priority
  - Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)
  - Bits[11:0]: VLAN Identifier (VID) field of VLAN tag
- When the ETV bit is set, only the VID is used for comparison.
- If this field ([11:0] if ETV is set) is all zeros, the MAC does not check the 15th and 16th bytes for VLAN tag comparison and declares all packets with Type field value of 0x8100 or 0x88a8 as VLAN packets.
VLAN Hash table register (ETH_MACVHTR)

Address offset: 0x0058
Reset value: 0x0000 0000

When the VTHM bit of **VLAN tag register (ETH_MACVTR)** register is set, the 16-bit VLAN Hash Table register is used for group address filtering based on the VLAN tag. For Hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on the ETV bit of **VLAN tag register (ETH_MACVTR)** register) in the incoming packet is passed through the CRC logic. The upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a Hash value of 1000 selects Bit 8 of the VLAN Hash table.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the VLAN tag or ID (For steps to calculate CRC32, see Section 3.2.8 of IEEE 802.3).
2. Perform bitwise reversal for the value obtained in step 1.
3. Take the upper four bits from the value obtained in step 2.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **VLHT[15:0]**: VLAN Hash Table

This field contains the 16-bit VLAN Hash Table.
The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the Transmit packets. It also contains the VLAN tag insertion controls.

### VLAN inclusion register (ETH_MACVIR)

**Address offset:** 0x0060  
**Reset value:** 0x0000 0000

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the Transmit packets. It also contains the VLAN tag insertion controls.

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-21</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

**Bit 20 VLTI:** VLAN Tag Input  
When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor.

**Bit 19 CSVL:** C-VLAN or S-VLAN  
When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.  
0: C-LAN  
1: S-LAN

**Bit 18 VLP:** VLAN Priority Control  
When this bit is set, the control bits[17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, bits[17:16] are ignored.

**Bits 17:16 VLC[1:0]:** VLAN Tag Control in Transmit Packets  
- 00: No VLAN tag deletion, insertion, or replacement  
- 01: VLAN tag deletion. The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted packets with VLAN tags.  
- 10: VLAN tag insertion. The MAC inserts VLT in bytes 15 and 16 of the packet after inserting the Type value (0x8100 or 0x88A8) in bytes 13 and 14. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.  
- 11: VLAN tag replacement. The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted packets (Bytes 13 and 14 are 0x8100 or 0x88A8).  

*Note:* Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

**Bits 15:0 VLT[15:0]:** VLAN Tag for Transmit Packets  
This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.  
The following list describes the bits of this field:  
- Bits[15:13]: User Priority  
- Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)  
- Bits[11:0]: VLAN Identifier (VID) field of VLAN tag
Inner VLAN inclusion register (ETH_MACIVIR)

Address offset: 0x0064
Reset value: 0x0000 0000

The Inner VLAN Tag Inclusion or Replacement register contains the inner VLAN tag to be inserted or replaced in the Transmit packet. It also contains the inner VLAN tag insertion controls.

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 |  1 |  0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 VLTI: VLAN Tag Input
When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor

Bit 19 CSVL: C-VLAN or S-VLAN
When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.
0: C-LAN
1: S-LAN

Bit 18 VLP: VLAN Priority Control
When this bit is set, the VLC field is used for VLAN deletion, insertion, or replacement. When this bit is reset, the VLC field is ignored.

Bits 17:16 VLC[1:0]: VLAN Tag Control in Transmit Packets
00: No VLAN tag deletion, insertion, or replacement
01: VLAN tag deletion
The MAC removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all transmitted packets with VLAN tags.
10: VLAN tag insertion
The MAC inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 17 and 18. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.
11: VLAN tag replacement
The MAC replaces VLT in bytes 19 and 20 of all VLAN-type transmitted packets (Bytes 17 and 18 are 0x8100 or 0x88a8).

Note: Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.
Bits 15:0 **VLT[15:0]**: VLAN Tag for Transmit Packets

This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.

The following list describes the bits of this field:
- Bits[15:13]: User Priority
- Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)
- Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

**Tx Queue flow control register (ETH_MACQTXFCR)**

Address offset: 0x0070

Reset value: 0x0000 0000

The Flow Control register controls the generation and reception of the Control (Pause Command) packets by the Flow control module of the MAC. A Write to a register with the Busy bit set to 1 triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register.

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th><strong>PT[15:0]</strong>: Pause Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>This field holds the value to be used in the Pause Time field in the Tx control packet.</td>
<td></td>
</tr>
<tr>
<td>Bits 15:8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

**Bit 7** **DZPQ**: Disable Zero-Quanta Pause

When this bit is set, it disables the automatic generation of the zero-quanta Pause packets. When this bit is reset, normal operation with automatic zero-quanta Pause packet generation is enabled.
Bits 6:4  **PLT[2:0]: Pause Low Threshold**

This field configures the threshold of the Pause timer at which the input flow is checked for automatic retransmission of the Pause packet.

The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted at 228 (256-28) slot times after the first Pause packet is transmitted.

The following list provides the threshold values for different values:

- **000**: Pause Time minus 4 Slot Times (PT-4 slot times)
- **001**: Pause Time minus 28 Slot Times (PT-28 slot times)
- **010**: Pause Time minus 36 Slot Times (PT-36 slot times)
- **011**: Pause Time minus 144 Slot Times (PT-144 slot times)
- **100**: Pause Time minus 256 Slot Times (PT-256 slot times)
- **101**: Pause Time minus 512 Slot Times (PT-512 slot times)
- **110 to 111**: Reserved, must not be used

The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the MII interface. This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.

Bits 3:2  Reserved, must be kept at reset value.

Bit 1  **TFE: Transmit Flow Control Enable**

**Full-duplex mode:** when this bit is set, the MAC enables the flow control operation to Tx Pause packets. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets.

**Half-duplex mode:** when this bit is set, the MAC enables the backpressure operation. When this bit is reset, the backpressure feature is disabled.

Bit 0  **FCB_BPA: Flow Control Busy or Backpressure Activate**

This bit initiates a Pause packet in the Full-duplex mode and activates the backpressure function in the Half-duplex mode if the TFE bit is set.

**Full-Duplex mode:** this bit should be read as 0 before writing to this register. To initiate a Pause packet, the application must set this bit to 1. During Control packet transfer, this bit continues to be set to indicate that a packet transmission is in progress. When Pause packet transmission is complete, the MAC resets this bit to 0. You should not write to this register until this bit is cleared.

**Half-duplex mode:** When this bit is set (and TFE bit is set) in the Half-duplex mode, the MAC asserts the backpressure. During backpressure, when the MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. When the MAC is configured for the Full-duplex mode, the BPA is automatically disabled.
Rx flow control register (ETH_MACRXFCR)

Address offset: 0x0090
Reset value: 0x0000 0000

The Receive Flow Control register controls the pausing of MAC Transmit based on the received Pause packet.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:2  Reserved, must be kept at reset value.

Bit 1  **UP**: Unicast Pause Packet Detect

A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When this bit is set, the MAC can also detect Pause packets with unicast address of the station. This unicast address should be as specified in **MAC Address 0 high register (ETH_MACA0HR)** and MAC Address 0 low register **MAC Address x low register (ETH_MACAxLR)**.

When this bit is reset, the MAC only detects Pause packets with unique multicast address.

**Note:** The MAC does not process a Pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01_80_C2_00_00_01) is as specified in IEEE 802.1 Qbb-2011.

Bit 0  **RFE**: Receive Flow Control Enable

When this bit is set and the MAC is operating in Full-duplex mode, the MAC decodes the received Pause packet and disables its transmitter for a specified (Pause) time. When this bit is reset or the MAC is operating in Half-duplex mode, the decode function of the Pause packet is disabled.

When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the Transmit queue, with matching priorities, for a duration of received Pause time.
Interrupt status register (ETH_MACISR)

Address offset: 0x00B0
Reset value: 0x0000 0000

The Interrupt Status register contains the status of interrupts.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:15  Reserved, must be kept at reset value.

Bit 14  RXSTSIS: Receive Status Interrupt
This bit indicates the status of received packets. This bit is set when the RWT bit is set in the Rx Tx status register (ETH_MACRXTXSR). This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of CSR software control register (ETH_MACCSRSWCR) is set) in the ETH_MACISR register.

Bit 13  TXSTSIS: Transmit Status Interrupt
This bit indicates the status of transmitted packets. This bit is set when any of the following bits is set in the Rx Tx status register (ETH_MACRXTXSR):
- Excessive Collision (EXCOL)
- Late Collision (LCOL)
- Excessive Deferral (EXDEF)
- Loss of Carrier (LCARR)
- No Carrier (NCARR)
- Jabber Timeout (TJT)

This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of CSR software control register (ETH_MACCSRSWCR) is set) in the ETH_MACISR register.
Bit 12 **TSIS**: Timestamp Interrupt Status
If the Timestamp feature is enabled, this bit is set when any of the following conditions is true:
- The system time value is equal to or exceeds the value specified in the Target Time High and Low registers.
- There is an overflow in the Seconds register.
- The Target Time Error occurred, that is, programmed target time already elapsed.
If the Auxiliary Snapshot feature is enabled, this bit is set when the auxiliary snapshot trigger is asserted.
When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the Tx timestamp status nanoseconds register (ETH_MACTXTSSNR) and Tx timestamp status seconds register (ETH_MACTXTSSSR) registers.
When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the Tx timestamp status nanoseconds register (ETH_MACTXTSSNR) and Tx timestamp status seconds register (ETH_MACTXTSSSR) registers, for PTO generated Delay Request and Pdelay request packets.
This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of CSR software control register (ETH_MACCSRSWCR) is set) in the Timestamp status register (ETH_MACTSSR).

Bit 11 Reserved, must be kept at reset value.

Bit 10 **MMCTXIS**: MMC Transmit Interrupt Status
This bit is set high when an interrupt is generated in the MMC Tx interrupt register (ETH_MMC_TX_INTERRUPT). This bit is cleared when all bits in this interrupt register are cleared.

Bit 9 **MMCRXIS**: MMC Receive Interrupt Status
This bit is set high when an interrupt is generated in the MMC Rx interrupt register (ETH_MMC_RX_INTERRUPT). This bit is cleared when all bits in this interrupt register are cleared.

Bit 8 **MMCIS**: MMC Interrupt Status
This bit is set high when MMCTXIS or MMCRXIS is set high. This bit is cleared only when all these bits are low.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **LPIS**: LPI Interrupt Status
This bit is set for any LPI state entry or exit in the MAC Transmitter or Receiver. This bit is cleared when the TLPIEN bit of LPI control and status register (ETH_MACLCSR) is read.

Bit 4 **PMTIS**: PMT Interrupt Status
This bit is set when a Magic packet or Wake-on-LAN packet is received in the power-down mode (RWKPRCVD and MGKPRCVD bits in ETH_MACPCSR register). This bit is cleared when Bits[6:5] are cleared because of a Read operation to the PMT control status register (ETH_MACPCSR).

Bit 3 **PHYIS**: PHY Interrupt
This bit is set when rising edge is detected on the ETH_PHY_INTN input. This bit is cleared when this register is read.

Bits 2:0 Reserved, must be kept at reset value.
**Interrupt enable register (ETH_MACIER)**

Address offset: 0x00B4  
Reset value: 0x0000 0000

The Interrupt Enable register contains the masks for generating the interrupts.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:15 Reserved, must be kept at reset value.

- **Bit 14 RXSTSIE**: Receive Status Interrupt Enable  
  When this bit is set, it enables the assertion of the interrupt signal because of the setting of RXSTSIS bit in the Interrupt status register (ETH_MACISR).

- **Bit 13 TXSTSIE**: Transmit Status Interrupt Enable  
  When this bit is set, it enables the assertion of the interrupt signal because of the setting of TXSTSIS bit in the Interrupt status register (ETH_MACISR).

- **Bit 12 TSIE**: Timestamp Interrupt Enable  
  When this bit is set, it enables the assertion of the interrupt signal because of the setting of TSIS bit in Interrupt status register (ETH_MACISR).

Bits 11:6 Reserved, must be kept at reset value.

- **Bit 5 LPIIE**: LPI Interrupt Enable  
  When this bit is set, it enables the assertion of the interrupt signal because of the setting of LPIIS bit in Interrupt status register (ETH_MACISR).

- **Bit 4 PMTIE**: PMT Interrupt Enable  
  When this bit is set, it enables the assertion of the interrupt signal because of the setting of PMTIS bit in Interrupt status register (ETH_MACISR).

- **Bit 3 PHYIE**: PHY Interrupt Enable  
  When this bit is set, it enables the assertion of the interrupt signal because of the setting of PHYIS bit in Interrupt status register (ETH_MACISR).

Bits 2:0 Reserved, must be kept at reset value.
**Rx Tx status register (ETH_MACRXTXSR)**

Address offset: 0x00B8

Reset value: 0x0000 0000

The Receive Transmit Status register contains the Receive and Transmit Error status.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:9</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>8</td>
<td><strong>RWT</strong>: Receive Watchdog Timeout</td>
</tr>
</tbody>
</table>
|     | This bit is set when a packet with length greater than 2,048 bytes is received (10, 240 bytes when Jumbo Packet mode is enabled) and the WD bit is reset in the Operating mode configuration register (ETH_MACCR). This bit is set when a packet with length greater than 16,383 bytes is received and the WD bit is set in the Operating mode configuration register (ETH_MACCR).  
|     | Cleared on read (or write of 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set). |
| 7:6 | Reserved, must be kept at reset value. |
| 5   | **EXCOL**: Excessive Collisions  |
|     | When the DTXSTS bit is set in the Operating mode Register (ETH_MTLOMR), this bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the Operating mode configuration register (ETH_MACCR), this bit is set after the first collision and the packet transmission is aborted.  
|     | Cleared on read (or write of 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set). |
| 4   | **LCOL**: Late Collision  |
|     | When the DTXSTS bit is set in the Operating mode Register (ETH_MTLOMR), this bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including Preamble in MII mode.  
|     | This bit is not valid if the Underflow error occurs.  
|     | Cleared on read (or write of 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set). |
| 3   | **EXDEF**: Excessive Deferral  |
|     | When the DTXSTS bit is set in the Operating mode Register (ETH_MTLOMR) and the DC bit is set in the Operating mode configuration register (ETH_MACCR), this bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 when Jumbo packet is enabled).  
|     | Cleared on read (or write of 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set). |
Bit 2  **LCARR**: Loss of Carrier

When the DTXSTS bit is set in the Operating mode Register (ETH_MTLOMR), this bit indicates that the loss of carrier occurred during packet transmission, that is, the ETH_CRS signal was inactive for one or more transmission clock periods during packet transmission. This bit is valid only for packets transmitted without collision.

Cleared on read (or write of 1 when RCWE bit in CSR software control register (ETH_MACCSRWCR) is set).

Bit 1  **NCARR**: No Carrier

When the DTXSTS bit is set in the Operating mode Register (ETH_MTLOMR), this bit indicates that the carrier signal from the PHY is not present at the end of preamble transmission.

Cleared on read (or write of 1 when RCWE bit in CSR software control register (ETH_MACCSRWCR) is set).

Bit 0  **TJT**: Transmit Jabber Timeout

This bit indicates that the Transmit Jabber Timer expired which happens when the packet size exceeds 2,048 bytes (10,240 bytes when the Jumbo packet is enabled) and JD bit is reset in the Operating mode configuration register (ETH_MACCR). This bit is set when the packet size exceeds 16,383 bytes and the JD bit is set in the Operating mode configuration register (ETH_MACCR).

Cleared on read (or write of 1 when RCWE bit in CSR software control register (ETH_MACCSRWCR) is set).

**PMT control status register (ETH_MACPCSR)**

Address offset: 0x00C0
Reset value: 0x0000 0000

| Bit 31 | RWKFILTRST: Remote wake-up Packet Filter Register Pointer Reset |
| Bit 30-29 | Reserved, must be kept at reset value. |
| Bit 28-24 | RWKPTR[4:0]: Remote wake-up FIFO Pointer |
| Bit 23-11 | Reserved, must be kept at reset value. |

| Bit 15-1 | RWKPKTEN: MGKPKTEN |
| Bit 8-7 | RWMPRVD: MGKPRVD |
| Bit 4-3 | RWKPRCVD |
| Bit 2-1 | RWKPFE |
| Bit 0 | GLBLCAST |

### RWKFILTRST

When this bit is set, the remote wake-up packet filter register pointer is reset to 0. It is automatically cleared after 1 clock cycle.

Bits 30:29  Reserved, must be kept at reset value.

Bits 28:24  **RWKPTR[4:0]**: Remote wake-up FIFO Pointer

This field gives the current value (0 to 7) of the Remote wake-up Packet Filter register pointer. When the value of this pointer is equal to 7, the contents of the Remote wake-up Packet Filter Register are transferred to the eth_mii_rx_clk domain when a Write occurs to that register.

Bits 23:11  Reserved, must be kept at reset value.
Bit 10  **RWKPF:** Remote wake-up Packet Forwarding Enable
When this bit is set along with RWKPKTEN, the MAC receiver drops all received frames until it receives the expected wake-up frame. All frames after that event including the received wake-up frame are forwarded to application. This bit is then self-cleared on receiving the wake-up packet.

The application can also clear this bit before the expected wake-up frame is received. In such cases, the MAC reverts to the default behavior where packets received are forwarded to the application. This bit must only be set when RWKPKTEN is set high and PWRDWN is set low.

The setting of this bit has no effect when PWRDWN is set high.

*Note:* If Magic Packet Enable and wake-up Frame Enable are both set along with setting of this bit and Magic Packet is received prior to wake-up frame, this bit is self-cleared on receiving Magic Packet, the received Magic packet is dropped, and all frames after received Magic Packet are forwarded to application.

Bit 9  **GLBLUCAST:** Global Unicast
When this bit set, any unicast packet filtered by the MAC (DAF) address recognition is detected as a remote wake-up packet.

Bits 8:7  Reserved, must be kept at reset value.

Bit 6  **RWKPRCVD:** Remote wake-up Packet Received
When this bit is set, it indicates that the power management event is generated because of the reception of a remote wake-up packet. This bit is cleared when this register is read.

Bit 5  **MGKPRCVD:** Magic Packet Received
When this bit is set, it indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared when this register is read (or written to 1 when RCWE bit in CSR software control register (ETH_MACCSR_SWCR) is set).

Bits 4:3  Reserved, must be kept at reset value.

Bit 2  **RWKPKTEN:** Remote wake-up Packet Enable
When this bit is set, a power management event is generated when the MAC receives a remote wake-up packet.

Bit 1  **MGKPKTEN:** Magic Packet Enable
When this bit is set, a power management event is generated when the MAC receives a magic packet.

Bit 0  **PWRDWN:** Power Down
When this bit is set, the MAC receiver drops all received packets until it receives the expected magic packet or remote wake-up packet. This bit is then self-cleared and the power-down mode is disabled. The software can clear this bit before the expected magic packet or remote wake-up packet is received. The packets received by the MAC after this bit is cleared are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Remote wake-up Packet Enable bit is set high.

*Note:* You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit.
Remote wake-up packet filter register (ETH_MACRWKPFR)

Address offset: 0x00C4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0 MACRWKPFR[31:0]: Remote wake-up packet filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refer to Table 649, Table 650 and Table 651 for details on register content and programming sequence.</td>
</tr>
</tbody>
</table>

The ETH_MACRWKPFR register at address 0x00C4 loads the wake-up Packet Filter register.

To load values in a wake-up Packet Filter register, the entire register (ETH_MACRWKPFR) must be written. The ETH_MACRWKPFR register is loaded by sequentially loading the eight, sixteen or thirty two register values in address (0x00C4) for ETH_MACRWKPFR value 0 to 7, respectively. The ETH_MACRWKPFR register is read in a similar way. The Ethernet peripheral updates the ETH_MACRWKPFR register current pointer value in Bits[26:24] of ETH_MACPCSR register.

LPI control and status register (ETH_MACLCSR)

Address offset: 0x00D0
Reset value: 0x0000 0000

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.
RM0481 Ethernet (ETH): media access control (MAC) with DMA controller

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **LPITCSE**: LPI Tx Clock Stop Enable

When this bit is set, the MAC asserts sdb_tck_gating_ctrl_o signal high after it enters Tx LPI mode to indicate that the Tx clock to MAC can be stopped. When this bit is reset, the MAC does not assert sdb_tck_gating_ctrl_o signal high after it enters Tx LPI mode.

If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern. The Tx Clock cannot be gated and so the LPITCSE bit cannot be programmed.

Bit 20 **LPITE**: LPI Timer Enable

This bit controls the automatic entry of the MAC Transmitter into and exit out of the LPI state. When LPITE, LPITXA and LPIEN bits are set, the MAC Transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the ETH_MACLETR register.

After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the Transmitter exits LPI state but does not clear LPIEN bit. This enables the re-entry into LPI state when it is IDLE again.

When LPITE is 0, the LPI Auto timer is disabled and MAC Transmitter enters LPI state based on the settings of LPITXA and LPIEN bit descriptions.

Bit 19 **LPITXA**: LPI Tx Automate

This bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the Transmit side.

If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding packets (in the core) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO Flush is set in the FTQ bit of ETH_MTLTxFOMR, when the MAC is in the LPI mode, it exits the LPI mode.

When this bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PLS**: PHY Link Status

This bit indicates the link status of the PHY. The MAC Transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER.

When this bit is set, the link is considered to be okay (UP) and when this bit is reset, the link is considered to be down.

Bit 16 **LPIEN**: LPI Enable

When this bit is set, it instructs the MAC Transmitter to enter the LPI state. When this bit is reset, it instructs the MAC to exit the LPI state and resume normal transmission.

This bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **RLPIST**: Receive LPI State

When this bit is set, it indicates that the MAC is receiving the LPI pattern on the MII interface.

Bit 8 **TLPIST**: Transmit LPI State

When this bit is set, it indicates that the MAC is transmitting the LPI pattern on the MII interface.

Bits 7:4 Reserved, must be kept at reset value.
Bit 3  **RLPIEX**: Receive LPI Exit

When this bit is set, it indicates that the MAC Receiver has stopped receiving the LPI pattern on the MII interface, exited the LPI state, and resumed the normal reception. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set).

*Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.*

Bit 2  **RLPIEN**: Receive LPI Entry

When this bit is set, it indicates that the MAC Receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set).

*Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.*

Bit 1  **TLPIEX**: Transmit LPI Exit

When this bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW Timer has expired. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set).

Bit 0  **TLPIEN**: Transmit LPI Entry

When this bit is set, it indicates that the MAC Transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set).

### LPI Timers Control Register (ETH_MACLTCR)

**Address offset**: 0x00D4

**Reset value**: 0x03E8 0000

The LPI Timers Control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-24</td>
<td>LST[9:0]</td>
<td>LPI Transmit Timer (TX)</td>
</tr>
<tr>
<td>15-12</td>
<td>TWT[15:0]</td>
<td>LPI Receive Timer (RX)</td>
</tr>
</tbody>
</table>

The LPI Timers Control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.
RM0481 Ethernet (ETH): media access control (MAC) with DMA controller

Bits 31:26  Reserved, must be kept at reset value.

Bits 25:16  **LST[9:0]:** LPI LS Timer
This field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programmed terminal count. The default value of the LPI LS Timer is 1000 (1 sec) as defined in the IEEE standard.

Bits 15:0  **TWT[15:0]:** LPI TW Timer
This field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPiEX status bit is set after the expiry of this timer.

**LPI entry timer register (ETH_MACLETR)**
Address offset: 0x00D8
Reset value: 0x0000 0000
This register controls the Tx LPI entry timer. This counter is enabled only when LPITE bit of LPI control and status register (ETH_MACLCSR) register is set to 1.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  **LPIET[19:0]:** LPI Entry Timer
This field specifies the time in microseconds the MAC waits after it has transmitted all the frames. This field is valid and used only when LPITE and LPITXA are set to 1.

Bits [2:0] are read-only so that the granularity of this timer is in steps of 8 micro-seconds.

**One-microsecond-tick counter register (ETH_MAC1USTCR)**
Address offset: 0x00DC
Reset value: 0x0000 0000
This register controls the generation of the Reference time (one-microsecond tick) for all the LPI timers. This timer has to be programmed by the software initially.

| Bit 31 | Bit 30 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9  | Bit 8  | Bit 7  | Bit 6  | Bit 5  | Bit 4  | Bit 3  | Bit 2  | Bit 1  | Bit 0  |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     |
| 15     | 14     | 13     | 12     | 11     | 10     | 9      | 8      | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      | 1      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
Bits 31:12  Reserved, must be kept at reset value.

Bits 11:0  TIC_1US_CNTR[11:0]: 1 µs tick Counter

The application must program this counter so that the number of clock cycles of CSR clock is 1 µs (subtract 1 from the value before programming).

For example if the CSR clock is 100 MHz then this field needs to be programmed to 100 - 1 = 99 (which is 0x63).

This is required to generate the 1 µs events that are used to update some of the EEE related counters.

**Version register (ETH_MACVR)**

Address offset: 0x0110

Reset value: 0x0000 3242

The version register identifies the version of the Ethernet peripheral.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>USERVER[7:0]</th>
<th>SNPSVER[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r</td>
<td>r r r r r r r</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:8  USERVER[7:0]: ST-defined version

Bits 7:0  SNPSVER[7:0]: IP version

**Debug register (ETH_MACDR)**

Address offset: 0x0114

Reset value: 0x0000 0000

The Debug register provides the debug status of various MAC blocks.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>RFCSTS[1:0]</th>
<th>RPESTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r</td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:19  Reserved, must be kept at reset value.

Bits 18:17  **TFCSTS[1:0]:** MAC Transmit Packet Controller Status

This field indicates the state of the MAC Transmit Packet Controller module:

00: Idle state
01: Waiting for one of the following:
   - Status of the previous packet
   - IPG or backoff period to be over
10: Generating and transmitting a Pause control packet (in Full-duplex mode)
11: Transferring input packet for transmission

Bit 16  **TPESTS:** MAC MII Transmit Protocol Engine Status

When this bit is set, it indicates that the MAC MII transmit protocol engine is actively transmitting data, and it is not in the Idle state.

Bits 15:3  Reserved, must be kept at reset value.

Bits 2:1  **RFCFCSTS[1:0]:** MAC Receive Packet Controller FIFO Status

This field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Packet Controller module.

Bit 0  **RPESTS:** MAC MII Receive Protocol Engine Status

When this bit is set, it indicates that the MAC MII receive protocol engine is actively receiving data, and it is not in the Idle state.

**HW feature 0 register (ETH_MACHWF0R)**

Address offset: 0x011C

Reset value: 0x0A0D 73F7

This register indicates the presence of first set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.
Bit 31  Reserved, must be kept at reset value.

Bits 30:28  **ACTPHYSSEL[2:0]:** Active PHY Selected
   When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion:
   000: GMII or MII
   001: RGMII
   010: SGMII
   011: TBI
   100: RMII
   101: RTBI
   110: SMII
   Others: Reserved, must not be used

Bit 27  **SAVLININS:** Source Address or VLAN Insertion Enable
   This bit is set to 1 when the Enable SA and VLAN Insertion on Tx option is selected

Bits 26:25  **TSSTSSSEL[1:0]:** Timestamp System Time Source
   This bit indicates the source of the Timestamp system time:
   01: Internal
   10: External
   11: Both
   00: Reserved, must not be used
   This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected

Bit 24  **MACADR64SEL:** MAC Addresses 64-127 Selected
   This bit is set to 1 when the Enable Additional 64 MAC Address Registers (64-127) option is selected

Bit 23  **MACADR32SEL:** MAC Addresses 32-63 Selected
   This bit is set to 1 when the Enable Additional 32 MAC Address Registers (32-63) option is selected

Bits 22:18  **ADDMACADRSEL[4:0]:** MAC Addresses 1-31 Selected
   This bit is set to 1 when the Enable Additional 1-31 MAC Address Registers option is selected

Bit 17  Reserved, must be kept at reset value.

Bit 16  **RXCOESEL:** Receive Checksum Offload Enabled
   This bit is set to 1 when the Enable Receive TCP/IP Checksum Check option is selected

Bit 15  Reserved, must be kept at reset value.

Bit 14  **TXCOESEL:** Transmit Checksum Offload Enabled
   This bit is set to 1 when the Enable Transmit TCP/IP Checksum Insertion option is selected

Bit 13  **EEESEL:** Energy Efficient Ethernet Enabled
   This bit is set to 1 when the Enable Energy Efficient Ethernet (EEE) option is selected

Bit 12  **TSSEL:** IEEE 1588-2008 Timestamp Enabled
   This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected

Bits 11:10  Reserved, must be kept at reset value.

Bit 9  **ARPOFFSEL:** ARP Offload Enabled
   This bit is set to 1 when the Enable IPv4 ARP Offload option is selected

Bit 8  **MMCSEL:** RMON Module Enable
   This bit is set to 1 when the Enable MAC management counters (MMC) option is selected
Bit 7  **MGKSEL**: PMT Magic Packet Enable
This bit is set to 1 when the Enable Magic Packet Detection option is selected

Bit 6  **RWKSEL**: PMT Remote wake-up Packet Enable
This bit is set to 1 when the Enable Remote wake-up Packet Detection option is selected

Bit 5  **SMASEL**: SMA (MDIO) Interface
This bit is set to 1 when the Enable Station Management (MDIO Interface) option is selected

Bit 4  **VLHASH**: VLAN Hash Filter Selected
This bit is set to 1 when the Enable VLAN Hash Table Based Filtering option is selected

Bit 3  **PCSSEL**: PCS Registers (TBI, SGMII, or RTBI PHY interface)
This bit is set to 1 when the TBI, SGMII, or RTBI PHY interface option is selected

Bit 2  **HDSEL**: Half-duplex Support
This bit is set to 1 when the Half-duplex mode is selected

Bit 1  **GMISEL**: 1000 Mbps Support
This bit is set to 1 when 1000 Mbps is selected as operating mode.

Bit 0  **MIISEL**: 10 or 100 Mbps Support
This bit is set to 1 when 10/100 Mbps is selected as operating mode.

**HW feature 1 register (ETH_MACHWF1R)**

Address offset: 0x0120
Reset value: 0x1104 1904

This register indicates the presence of second set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.
Bit 31  Reserved, must be kept at reset value.

Bits 30:27  **L3L4FNUM[3:0]**: Total number of L3 or L4 Filters
           This field indicates the total number of L3 or L4 filters:
           0000: No L3 or L4 Filter
           0001: 1 L3 or L4 Filter
           0010: 2 L3 or L4 Filters
           ..
           1000: 8 L3 or L4

Bit 26  Reserved, must be kept at reset value.

Bits 25:24  **HASHTBLSZ[1:0]**: Hash Table Size
           This field indicates the size of the Hash table:
           00: No Hash table
           01: 64
           10: 128
           11: 256

Bit 23  **POUST**: One Step for PTP over UDP/IP Feature Enable
           This bit is set to 1 when the Enable one step timestamp for PTP over UDP/IP feature is selected.

Bit 22  Reserved, must be kept at reset value.

Bit 21  **RAVSEL**: Rx Side Only AV Feature Enable
           This bit is set to 1 when the Enable Audio video bridging option on Rx Side Only is selected.

Bit 20  **AVSEL**: AV Feature Enable
           This bit is set to 1 when the Enable Audio video bridging option is selected.

Bit 19  **DBGMEMA**: DMA Debug Registers Enable
           This bit is set to 1 when the Debug Mode Enable option is selected

Bit 18  **TSOEN**: TCP Segmentation Offload Enable
           This bit is set to 1 when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected

Bit 17  **SPHEN**: Split Header Feature Enable
           This bit is set to 1 when the Enable Split Header Structure option is selected

Bit 16  **DCBEN**: DCB Feature Enable
           This bit is set to 1 when the Enable Data Center Bridging option is selected

Bits 15:14  **ADDR64[1:0]**: Address width
           This field indicates the configured address width.
           00: 32 bits
           Others: Reserved, must not be used

Bit 13  **ADVTHWORD**: IEEE 1588 High Word Register Enable
           This bit is set to 1 when the Add IEEE 1588 Higher Word Register option is selected

Bit 12  **PTOEN**: PTP Offload Enable
           This bit is set to 1 when the Enable PTP Timestamp Offload Feature is selected.

Bit 11  **OSTEN**: One-Step Timestamping Enable
           This bit is set to 1 when the Enable One-Step Timestamp Feature is selected.
Bits 10:6 **TXFIFOSIZE[4:0]**: MTL Transmit FIFO Size

This field contains the configured value of MTL Tx FIFO in bytes expressed as \( \log_2 \text{TXFIFO\_SIZE} - 7 \):

- 00000: 128 bytes
- 00001: 256 bytes
- 00010: 512 bytes
- 00011: 1,024 bytes
- 00100: 2,048 bytes
- 00101: 4,096 bytes
- 00110: 8,192 bytes
- 00111: 16,384 bytes
- 01000: 32 Kbytes
- 01001: 64 Kbytes
- 01010: 128 Kbytes
- 01011 to 11111: Reserved, must not be used

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **RXFIFOSIZE[4:0]**: MTL Receive FIFO Size

This field contains the configured value of MTL Rx FIFO in bytes expressed as \( \log_2 \text{RXFIFO\_SIZE} - 7 \):

- 00000: 128 bytes
- 00001: 256 bytes
- 00010: 512 bytes
- 00011: 1,024 bytes
- 00100: 2,048 bytes
- 00101: 4,096 bytes
- 00110: 8,192 bytes
- 00111: 16,384 bytes
- 01000: 32 Kbytes
- 01001: 64 Kbytes
- 01010: 128 Kbytes
- 01011 to 11111: Reserved, must not be used
HW feature 2 register (ETH_MACHWF2R)

Address offset: 0x0124

Reset value: 0x4100 0000

This register indicates the presence of third set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Bit 31  Reserved, must be kept at reset value.

Bits 30:28  **AUXSNAPNUM[2:0]**: Number of Auxiliary Snapshot Inputs

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>No auxiliary input</td>
</tr>
<tr>
<td>001</td>
<td>1 auxiliary input</td>
</tr>
<tr>
<td>010</td>
<td>2 auxiliary inputs</td>
</tr>
<tr>
<td>011</td>
<td>3 auxiliary inputs</td>
</tr>
<tr>
<td>100</td>
<td>4 auxiliary inputs</td>
</tr>
<tr>
<td>101 to 111</td>
<td>Reserved, must not be used</td>
</tr>
</tbody>
</table>

Bit 27  Reserved, must be kept at reset value.

Bits 26:24  **PPSOUTNUM[2:0]**: Number of PPS Outputs

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>No PPS output</td>
</tr>
<tr>
<td>001</td>
<td>1 PPS output</td>
</tr>
<tr>
<td>010</td>
<td>2 PPS outputs</td>
</tr>
<tr>
<td>011</td>
<td>3 PPS outputs</td>
</tr>
<tr>
<td>100</td>
<td>4 PPS outputs</td>
</tr>
<tr>
<td>101 to 111</td>
<td>Reserved, must not be used</td>
</tr>
</tbody>
</table>

Bits 23:22  **TDCSZ[1:0]**: Tx DMA Descriptor Cache Size in terms of 16-byte descriptors

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Cache not configured</td>
</tr>
<tr>
<td>01</td>
<td>Four 16-byte descriptors</td>
</tr>
<tr>
<td>10</td>
<td>Eight 16-byte descriptors</td>
</tr>
<tr>
<td>11</td>
<td>Sixteen 16-byte descriptors</td>
</tr>
</tbody>
</table>

Bits 21:18  **TXCHCNT[3:0]**: Number of DMA Transmit Channels

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1 DMA Tx Channel</td>
</tr>
<tr>
<td>0001</td>
<td>2 DMA Tx Channels</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0111</td>
<td>8 DMA Tx</td>
</tr>
</tbody>
</table>
Bits 17:16  **RDCSZ[1:0]**: Rx DMA Descriptor Cache Size in terms of 16-byte descriptors
- 00: Cache not configured
- 01: Four 16-byte descriptors
- 10: Eight 16-byte descriptors
- 11: Sixteen 16-byte descriptors

Bits 15:12  **RXCHCNT[3:0]**: Number of DMA Receive Channels
This field indicates the number of DMA Receive channels:
- 0000: 1 DMA Rx Channel
- 0001: 2 DMA Rx Channels
- ..
- 0111: 8 DMA Rx

Bits 11:10  Reserved, must be kept at reset value.

Bits 9:6  **TXQCNT[3:0]**: Number of MTL Transmit Queues
This field indicates the number of MTL Transmit queues:
- 0000: 1 MTL Tx queue
- 0001: 2 MTL Tx queues
- ..
- 0111: 8 MTL Tx

Bits 5:4  Reserved, must be kept at reset value.

Bits 3:0  **RXQCNT[3:0]**: Number of MTL Receive Queues
This field indicates the number of MTL Receive queues:
- 0000: 1 MTL Rx queue
- 0001: 2 MTL Rx queues
- ..
- 0111: 8 MTL Rx
HW feature 3 register (ETH_MACHWF3R)

Address offset: 0x0128
Reset value: 0x0000 0020

This register indicates the presence of fourth set the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rsx</td>
<td>Rsx</td>
<td>Rsx</td>
<td>Rsx</td>
<td>Rsx</td>
<td>Rsx</td>
<td>Rsx</td>
<td>Rsx</td>
<td>Rsx</td>
<td>Rsx</td>
<td>Rsx</td>
<td>Rsx</td>
<td>Rsx</td>
<td>Rsx</td>
<td>Rsx</td>
<td>Rsx</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>

Bits 31:6  Reserved, must be kept at reset value.

Bit 5  **DVLAN**: Double VLAN processing enable
This bit is set to 1 when Double VLAN processing is enabled.

Bit 4  **CBTISEL**: Queue/Channel based VLAN tag insertion on Tx enable
This bit is set to 1 when the Enable Queue/Channel based VLAN tag insertion on Tx feature is selected.

Bit 3  Reserved, must be kept at reset value.

Bits 2:0  **NRVF[2:0]**: Number of Extended VLAN Tag Filters Enabled
This field indicates the Number of Extended VLAN Tag Filters selected:
- 000: No Extended Rx VLAN Filters
- 001: 4 Extended Rx VLAN Filters
- 010: 8 Extended Rx VLAN Filters
- 011: 16 Extended Rx VLAN Filters
- 100: 24 Extended Rx VLAN Filters
- 101: 32 Extended Rx VLAN Filters
- 110 to 111: Reserved, must not be used
MDIO address register (ETH_MACMDIOAR)

Address offset: 0x0200
Reset value: 0x0000 0000

The MDIO Address register controls the management cycles to external PHY through a management interface.

<table>
<thead>
<tr>
<th>Bit 31:28</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 27</td>
<td>PSE: Preamble Suppression Enable</td>
</tr>
<tr>
<td></td>
<td>When this bit is set, the SMA suppresses the 32-bit preamble and transmit MDIO frames with only 1 preamble bit.</td>
</tr>
<tr>
<td></td>
<td>When this bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications.</td>
</tr>
<tr>
<td>Bit 26</td>
<td>BTB: Back to Back transactions</td>
</tr>
<tr>
<td></td>
<td>When this bit is set and the NTC has value greater than 0, then the MAC informs the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which is executed immediately irrespective of the number trailing clocks generated for the previous frame.</td>
</tr>
<tr>
<td></td>
<td>When this bit is reset, then the read/write command completion (MII busy is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame.</td>
</tr>
<tr>
<td></td>
<td>This bit must not be set when NTC=0.</td>
</tr>
<tr>
<td>Bits 25:21</td>
<td>PA[4:0]: Physical Layer Address</td>
</tr>
<tr>
<td></td>
<td>This field indicates which Clause 22 PHY devices (out of 32 devices) the MAC is accessing.</td>
</tr>
<tr>
<td></td>
<td>This field indicates which Clause 45 capable PHYs (out of 32 PHYs) the MAC is accessing.</td>
</tr>
<tr>
<td>Bits 20:16</td>
<td>RDA[4:0]: Register/Device Address</td>
</tr>
<tr>
<td></td>
<td>These bits select the PHY register in selected Clause 22 PHY device. These bits select the Device (MMD) in selected Clause 45 capable PHY.</td>
</tr>
<tr>
<td>Bit 15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 14:12</td>
<td>NTC[2:0]: Number of Training Clocks</td>
</tr>
<tr>
<td></td>
<td>This field controls the number of trailing clock cycles generated on ETH_MDC after the end of transmission of MDIO frame. The valid values can be from 0 to 7. Programming the value to 011 indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer.</td>
</tr>
</tbody>
</table>
Bits 11:8 **CR[3:0]: CSR Clock Range**

The CSR Clock Range selection determines the frequency of the MDC clock according to the CSR clock frequency used in your design:

- **0000**: CSR clock = 60-100 MHz; MDC clock = CSR clock/42
- **0001**: CSR clock = 100-150 MHz; MDC clock = CSR clock/62
- **0010**: CSR clock = 20-35 MHz; MDC clock = CSR clock/16
- **0011**: CSR clock = 35-60 MHz; MDC clock = CSR clock/26
- **0100**: CSR clock = 150-250 MHz; MDC clock = CSR clock/102
- **0101**: CSR clock = 250-300 MHz; MDC clock = CSR clock/124
- **0110** to **0111**: Reserved, must not be used

The suggested range of CSR clock frequency applicable for each value (when Bit 11 = 0) ensures that the MDC clock is approximately between 1.0 MHz to 2.5 MHz frequency range. When Bit 11 is set, you can achieve a higher frequency of the MDC clock than the frequency limit of 2.5 MHz (specified in the IEEE 802.3) and program a clock divider of lower value. For example, when CSR clock is of 100 MHz frequency and you program these bits to 1010, the resultant MDC clock is of 12.5 MHz which is above the range specified in IEEE 802.3.

Program the following values only if the interfacing chips support faster MDC clocks:

- **1000**: CSR clock/4
- **1001**: CSR clock/6
- **1010**: CSR clock/8
- **1011**: CSR clock/10
- **1100**: CSR clock/12
- **1101**: CSR clock/14
- **1110**: CSR clock/16
- **1111**: CSR clock/18

Bits 7:5 **Reserved, must be kept at reset value.**

Bit 4 **SKAP**: Skip Address Packet

When this bit is set, the SMA does not send the address packets before read, write, or post-read increment address packets. This bit is valid only when C45E is set.
Bits 3:2 **GOC[1:0]:** MII Operation Command
This bit indicates the operation command to the PHY.
00: Reserved, must not be used
01: Write
10: Post Read Increment Address for Clause 45 PHY
11: Read
When Clause 22 PHY is enabled, only Write and Read commands are valid.

Bit 1 **C45E: Clause 45 PHY Enable**
When this bit is set, Clause 45 capable PHY is connected to MDIO. When this bit is reset, Clause 22 capable PHY is connected to MDIO.

Bit 0 **MB: MII Busy**
The application sets this bit to instruct the SMA to initiate a Read or Write access to the MDIOS. The MAC clears this bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in **MDIO address register (ETH_MACMDIOAR)** and **MDIO data register (ETH_MACMDIODR)** as long as this bit is set.

For write transfers, the application must first write 16-bit data in the MD field (and also RA field when C45E is set) in **MDIO data register (ETH_MACMDIODR)** register before setting this bit. When C45E is set, it should also write into the RA field of **MDIO data register (ETH_MACMDIODR)** before initiating a read transfer. When a read transfer is completed (MII busy=0), the data read from the PHY register is valid in the MD field of the **MDIO data register (ETH_MACMDIODR)**.

Note: *Even if the addressed PHY is not present, there is no change in the functionality of this bit.*

**MDIO data register (ETH_MACMDIODR)**
Address offset: 0x0204
Reset value: 0x0000 0000

The MDIO Data register stores the Write data to be written to the PHY register located at the address specified in **MDIO address register (ETH_MACMDIOAR)**. This register also stores the Read data from the PHY register located at the address specified by MDIO Address register.

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits 3:0</th>
<th>Bits 15:4</th>
<th>Bits 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>MD[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 **RA[15:0]:** Register Address
This field is valid only when C45E is set. It contains the Register Address in the PHY to which the MDIO frame is intended for.

Bits 15:0 **MD[15:0]:** MII Data
This field contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.
ARP address register (ETH_MACARPAR)

Address offset: 0x0210
Reset value: 0x0000 0000

The ARP Address register contains the IPv4 Destination Address of the MAC.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

ARPPA[31:16]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

ARPPA[15:0]

Bits 31:0 **ARPPA[31:0]**: ARP Protocol Address

This field contains the IPv4 Destination Address of the MAC. This address is used for perfect match with the Protocol Address of Target field in the received ARP packet.

CSR software control register (ETH_MACCSRSWCR)

Address offset: 0x0230
Reset value: 0x0000 0000

This register contains software-programmable controls for changing the CSR access response and status bits clearing.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

| Bits 31:9 Reserved, must be kept at reset value. |

Bit 8 **SEEN**: Slave Error Response Enable

When this bit is set, the MAC responds with a Slave Error for accesses to reserved registers in CSR space.
When this bit is reset, the MAC responds with an Okay response to any register accessed from CSR space.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **RCWE**: Register Clear on Write 1 Enable

When this bit is set, the access mode to some register fields changes to rc_w1 (clear on write) meaning that the application needs to set that respective bit to 1 to clear it.
When this bit is reset, the access mode to these register fields remains rc_r (clear on read).
MAC Address 0 high register (ETH_MACA0HR)

Address offset: 0x0300
Reset value: 0x8000 FFFF

The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the MII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the MII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x665544332211.

If the MAC address registers are configured to be double-synchronized to the MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>AE</td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **AE**: Address Enable
This bit is always set to 1.

Bits 30:16 **Reserved, must be kept at reset value.**

Bits 15:0 **ADDRHI[15:0]**: MAC Address0[47:32]
This field contains the upper 16 bits [47:32] of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

MAC Address x low register (ETH_MACAxLR)

Address offset: 0x0304 + 0x8 * x, (x = 0 to 3)
Reset value: 0xFFFF FFFF

The MAC Address x Low register holds the lower 32 bits of the 6-byte first MAC address of the station.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>ADDRLO[31:16]</td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>ADDRLO[15:0]</td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:0  **ADDRLO[31:0]**: MAC Address x [31:0] (x = 0 to 3)  
This field contains the lower 32 bits of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

**MAC Address x high register (ETH_MACAxHR)**

Address offset: 0x0308 + 0x8 * (x-1), (x = 1 to 3)

Reset value: 0x0000 FFFF

The MAC Address x High register holds the upper 16 bits of the second 6-byte MAC address of the station.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**ADDRHI[15:0]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31  **AE**: Address Enable  
When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

Bit 30  **SA**: Source Address  
When this bit is set, the MAC Addressx[47:0] is used to compare with the SA fields of the received packet. When this bit is reset, the MAC Address x[47:0] is used to compare with the DA fields of the received packet.  
0: DA  
1: SA

Bits 29:24  **MBC[5:0]**: Mask Byte Control  
These bits are mask control bits for comparing each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows:  
Bit 29: ETH_MACAxHR[15:8]  
Bit 28: ETH_MACAxHR[7:0]  
Bit 27: ETH_MACAxLR[31:24]  
Bit 26: ETH_MACAxLR[23:16]  
Bit 25: ETH_MACAxLR[15:8]  
Bit 24: ETH_MACAxLR[7:0]  
You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.

Bits 23:16  **Reserved**, must be kept at reset value.

Bits 15:0  **ADDRHI[15:0]**: MAC Address1 [47:32]  
This field contains the upper 16 bits[47:32] of the second 6-byte MAC address.
### MMC control register (ETH_MMC_CONTROL)

Address offset: 0x0700  
Reset value: 0x0000 0000

This register configures the MMC operating mode.

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UCDBC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CNTPRSTLVL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CNTPRST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CNTFREEZ</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:9  Reserved, must be kept at reset value.

Bit 8 **UCDBC**: Update MMC Counters for Dropped Broadcast Packets

The CNTRST bit has a higher priority than the CNTPRST bit. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the CNTPRST bit is not set.

When set, the MAC updates all related MMC Counters for Broadcast packets that are dropped because of the setting of the DBF bit of *Packet filtering control register (ETH_MACPFR)*.

When reset, the MMC Counters are not updated for dropped Broadcast packets.

Bits 7:6  Reserved, must be kept at reset value.

Bit 5 **CNTPRSTLVL**: Full-Half Preset

When this bit is low and the CNTPRST bit is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF_F800 (Half 2Kbytes) and all packet-counters get preset to 0x7FFF_FFF0 (Half 16).

When this bit is high and the CNTPRST bit is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0x7FFF_F800 (Half 2Kbytes) and all packet-counters get preset to 0x7FFF_FFF0 (Half 16).

For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and packet counters. Similarly, the almost-full preset values for the 16-bit counters are 0x800 and 0xFFFF.

Bit 4 **CNTPRST**: Counters Preset

When this bit is set, all counters are initialized or preset to almost full or almost half according to the CNTPRSTLVL bit. This bit is cleared automatically after 1 clock cycle.

This bit, along with the CNTPRSTLVL bit, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full.

Bit 3 **CNTFREEZ**: MMC Counter Freeze

When this bit is set, it freezes all MMC counters to their current value. Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received packet. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.
Bit 2 **RSTONRD**: Reset on Read  
When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset).  
The counters are cleared when the least significant byte lane (Bits[7:0]) is read.

Bit 1 **CNTSTOPRO**: Counter Stop Rollover  
When this bit is set, the counter does not roll over to zero after reaching the maximum value.

Bit 0 **CNTRST**: Counters Reset  
When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle.

**MMC Rx interrupt register (ETH_MMC_RX_INTERRUPT)**  
Address offset: 0x0704  
Reset value: 0x0000 0000  
This register maintains the interrupts generated from all Receive statistics counters.

The MMC Receive Interrupt register maintains the interrupts that are generated when the following occur:

- Receive statistic counters reach half of their maximum values (0x8000_0000 for 32 bit counter and 0x8000 for 16 bit counter).
- Receive statistic counters cross their maximum values (0xFFFF_FFFF for 32 bit counter and 0xFFFF for 16 bit counter).

When the CNTSTOPRO is set in **MMC control register (ETH_MMC_CONTROL)**, interrupts are set but the counter remains at all-ones. The MMC Receive Interrupt register is a 32 bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

<table>
<thead>
<tr>
<th>Bit 31:28</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bit 27 **RXLPIITRCIS**: MMC Receive LPI transition counter interrupt status  
This bit is set when the **Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)** counter reaches half of the maximum value or the maximum value. |
| Bit 26 **RXLPIUSCIS**: MMC Receive LPI microsecond counter interrupt status  
This bit is set when the **Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)** counter reaches half of the maximum value or the maximum value. |
| Bits 25:18 | Reserved, must be kept at reset value. |
Bit 17 **RXUCGPIS**: MMC Receive Unicast Good Packet Counter Interrupt Status

This bit is set when the *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 16-7 Reserved, must be kept at reset value.

Bit 6 **RXALGNERPIS**: MMC Receive Alignment Error Packet Counter Interrupt Status

This bit is set when the *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCEPIS**: MMC Receive CRC Error Packet Counter Interrupt Status

This bit is set when the *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 4:0 Reserved, must be kept at reset value.

**MMC Tx interrupt register (ETH_MMC_TX_INTERRUPT)**

Address offset: 0x0708

Reset value: 0x0000 0000

This register maintains the interrupts generated from all Transmit statistics counters.

The MMC Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x8000_0000 for 32 bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFF for 16-bit counter).

When CNTSTOPRO is set in *MMC control register (ETH_MMC_CONTROL)*, the interrupts are set but the counter remains at all-ones.

The MMC Transmit Interrupt register is a 32 bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.

The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rx_r</td>
<td>Rx_r</td>
<td>Rx_r</td>
<td>Rx_r</td>
<td>TXLPITRCIS</td>
<td>TXLPIUSCIS</td>
<td>RXGPKTIS</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXMCOLGPIS</td>
<td>RXSCOLGPIS</td>
<td>RX_r</td>
<td>RX_r</td>
<td>RX_r</td>
<td>RX_r</td>
<td>RX_r</td>
<td>RX_r</td>
<td>RX_r</td>
<td>RX_r</td>
<td>RX_r</td>
<td>RX_r</td>
<td>RX_r</td>
<td>RX_r</td>
<td>RX_r</td>
<td>RX_r</td>
</tr>
</tbody>
</table>

rc_r rc_r
Bits 31:28  Reserved, must be kept at reset value.

Bit 27  **TXLPIITRCIS**: MMC Transmit LPI transition counter interrupt status
This bit is set when the *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26  **TXLPIUSCIS**: MMC Transmit LPI microsecond counter interrupt status
This bit is set when the *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:22  Reserved, must be kept at reset value.

Bit 21  **TXGPKTIS**: MMC Transmit Good Packet Counter Interrupt Status
This bit is set when the *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 20:16  Reserved, must be kept at reset value.

Bit 15  **TXMCOLGPIIS**: MMC Transmit Multiple Collision Good Packet Counter Interrupt Status
This bit is set when the *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 14  **TXSCOLGPIIS**: MMC Transmit Single Collision Good Packet Counter Interrupt Status
This bit is set when the *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 13:0  Reserved, must be kept at reset value.
### MMC Rx interrupt mask register (ETH_MMC_RX_INTERRUPT_MASK)

Address offset: 0x070C  
Reset value: 0x0000 0000

The MMC Receive Interrupt Mask register maintains the masks for the interrupts generated when receive statistic counters reach half of their maximum value or the maximum values.

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Bits 31:28 | Reserved, must be kept at reset value.

**Bit 27** **RXLPITRCIM**: MMC Receive LPI transition counter interrupt Mask  
Setting this bit masks the interrupt when the Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR) counter reaches half of the maximum value or the maximum value.

**Bit 26** **RXLPIUSCIM**: MMC Receive LPI microsecond counter interrupt Mask  
Setting this bit masks the interrupt when the Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR) counter reaches half of the maximum value or the maximum value.

**Bits 25:18** Reserved, must be kept at reset value.

**Bit 17** **RXUCGPIM**: MMC Receive Unicast Good Packet Counter Interrupt Mask  
Setting this bit masks the interrupt when the Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD) counter reaches half of the maximum value or the maximum value.

**Bits 16:7** Reserved, must be kept at reset value.

**Bit 6** **RXALGNERPM**: MMC Receive Alignment Error Packet Counter Interrupt Mask  
Setting this bit masks the interrupt when the Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS) counter reaches half of the maximum value or the maximum value.

**Bit 5** **RXCRCERPIM**: MMC Receive CRC Error Packet Counter Interrupt Mask  
Setting this bit masks the interrupt when the Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS) counter reaches half of the maximum value or the maximum value.

**Bits 4:0** Reserved, must be kept at reset value.
MMC Tx interrupt mask register (ETH_MMC_TX_INTERRUPT_MASK)

Address offset: 0x0710
Reset value: 0x0000 0000

The MMC Transmit Interrupt Mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or the maximum values. This register is 32 bit wide.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXLPITRCIM**: MMC Transmit LPI transition counter interrupt Mask
Setting this bit masks the interrupt when the Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR) counter reaches half of the maximum value or the maximum value.

Bit 26 **TXLPIUSCIM**: MMC Transmit LPI microsecond counter interrupt Mask
Setting this bit masks the interrupt when the Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR) counter reaches half of the maximum value or the maximum value.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **TXGPKTIM**: MMC Transmit Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD) counter reaches half of the maximum value or the maximum value.

Bits 20:16 Reserved, must be kept at reset value.

Bit 15 **TXMCOLGPIM**: MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS) counter reaches half of the maximum value or the maximum value.

Bit 14 **TXSCOLGPIM**: MMC Transmit Single Collision Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS) counter reaches half of the maximum value or the maximum value.

Bits 13:0 Reserved, must be kept at reset value.
**Tx single collision good packets register**  
(ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)

Address offset: 0x074C  
Reset value: 0x0000 0000  
This register provides the number of successfully transmitted packets by Ethernet peripheral after a single collision in the Half-duplex mode.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **TXSNGLCOLG[31:0]**: Tx Single Collision Good Packets  
This field indicates the number of successfully transmitted packets after a single collision in the Half-duplex mode.

**Tx multiple collision good packets register**  
(ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)

Address offset: 0x0750  
Reset value: 0x0000 0000  
This register provides the number of successfully transmitted packets by Ethernet peripheral after multiple collisions in the Half-duplex mode.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **TXMULTCOLG[31:0]**: Tx Multiple Collision Good Packets  
This field indicates the number of successfully transmitted packets after multiple collisions in the Half-duplex mode.
**Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)**

Address offset: 0x0768  
Reset value: 0x0000 0000  
This register provides the number of good packets transmitted by Ethernet peripheral.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **TXPKTG[31:0]**: Tx Packet Count Good  
This field indicates the number of good packets transmitted.

**Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)**

Address offset: 0x0794  
Reset value: 0x0000 0000  
This register provides the number of packets received by Ethernet peripheral with CRC error.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **RXCRCERR[31:0]**: Rx CRC Error Packets  
This field indicates the number of packets received with CRC error.
Rx alignment error packets register
(ETH_RX_ALIGNMENT_ERROR_PACKETS)
Address offset: 0x0798
Reset value: 0x0000 0000
This register provides the number of packets received by Ethernet peripheral with alignment (dribble) error. It is valid only in 10/100 mode.

<table>
<thead>
<tr>
<th>RXALGNERR[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r     r     r     r     r     r     r     r     r     r     r     r</td>
</tr>
<tr>
<td>15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0</td>
</tr>
</tbody>
</table>

Bits 31:0 **RXALGNERR[31:0]: Rx Alignment Error Packets**
This field indicates the number of packets received with alignment (dribble) error. It is valid only in 10/100 mode.

Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)
Address offset: 0x07C4
Reset value: 0x0000 0000
This register provides the number of good unicast packets received by Ethernet peripheral.

<table>
<thead>
<tr>
<th>RXUCASTG[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r     r     r     r     r     r     r     r     r     r     r     r</td>
</tr>
<tr>
<td>15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RXUCASTG[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r     r     r     r     r     r     r     r     r     r     r     r</td>
</tr>
</tbody>
</table>

Bits 31:0 **RXUCASTG[31:0]: Rx Unicast Packets Good**
This field indicates the number of good unicast packets received.
### Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)

Address offset: 0x07EC
Reset value: 0x0000 0000

This register provides the number of microseconds Tx LPI is asserted by Ethernet peripheral.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TXLPUSC[31:16]**

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**TXLPUSC[15:0]**

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 **TXLPUSC[31:0]**: Tx LPI Microseconds Counter

This field indicates the number of microseconds Tx LPI is asserted. For every Tx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

### Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)

Address offset: 0x07F0
Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Tx LPI.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TXLPICTR[31:16]**

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**TXLPICTR[15:0]**

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 **TXLPICTR[31:0]**: Tx LPI Transition counter

This field indicates the number of times Tx LPI Entry has occurred. Even if Tx LPI Entry occurs in Automate mode (because of LPITXA bit set in the **LPI control and status register (ETH_MACLCSR)**), the counter is incremented.
Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)

Address offset: 0x07F4
Reset value: 0x0000 0000

This register provides the number of microseconds Rx LPI is sampled by Ethernet peripheral.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 RXLPIUSC[31:0]: Rx LPI Microseconds Counter
This field indicates the number of microseconds Rx LPI is asserted. For every Rx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)

Address offset: 0x07F8
Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Rx LPI.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 RXLPITRC[31:0]: Rx LPI Transition counter
This field indicates the number of times Rx LPI Entry has occurred.
**L3 and L4 control 0 register (ETH_MACL3L4C0R)**

Address offset: 0x0900  
Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 0 of Layer 3 and Layer 4.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31:22</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td><strong>L4DPM0</strong>: Layer 4 Destination Port Inverse Match Enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|      | When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching.  
This bit is valid and applicable only when the L4DPM0 bit is set high. |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 20   | **L4DPM0**: Layer 4 Destination Port Match Enable |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|      | When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching. |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 19   | **L4SPM0**: Layer 4 Source Port Inverse Match Enable |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|      | When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching.  
This bit is valid and applicable only when the L4SPM0 bit is set high. |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 18   | **L4SPM0**: Layer 4 Source Port Match Enable |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|      | When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching. |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 17   | Reserved, must be kept at reset value.   |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 16   | **L4PEN0**: Layer 4 Protocol Enable      |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|      | When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.  
The Layer 4 matching is done only when the L4SPM0 or L4DPM0 bit is set. |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
Bits 15:11 **L3HDBM0[4:0]**: Layer 3 IP DA higher bits match

Condition: IPv4 packets

This field contains the number of higher bits of IP Destination Address that are masked in the IPv4 packets:

0: No bits are masked.
1: Lsb[0] is masked
2: Two Lsbs [1:0] are masked
.. 
31: All bits except MSb are masked.

Condition: IPv6 packets

Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM0 which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. Number of bits masked is given by concatenated values of the L3HDBM0[1:0] and L3HSBM0 bits:

0: No bits are masked.
1: Lsb[0] is masked
2: Two Lsbs [1:0] are masked
.. 
31: All bits except MSb are masked.

This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set.

Bits 10:6 **L3HSBM0[4:0]**: Layer 3 IP SA higher bits match

Condition: IPv4 packets

This field contains the number of lower bits of IP source address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.
1: Lsb[0] is masked
2: Two Lsbs [1:0] are masked
.. 
31: All bits except MSb are masked.

Condition: IPv6 packets:

This field contains Bits[4:0] of L3HSBM0. These bits indicate the number of higher bits of IP source or destination address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set high.

Bit 5 **L3DAIM0**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM0 bit is set high.

Bit 4 **L3DAM0**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

*Note:* When the L3PEN0 bit is set, you should set either this bit or the L3SAM0 bit because either IPv6 DA or SA can be checked for filtering.

Bit 3 **L3SAIM0**: Layer 3 IP SA Inverse Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Source Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3SAM0 bit is set.
Bit 2  **L3SAM0**: Layer 3 IP SA Match Enable
When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

*Note:* When the L3PEN0 bit is set, you should set either this bit or the L3DAM0 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1  Reserved, must be kept at reset value.

Bit 0  **L3PEN0**: Layer 3 Protocol Enable
When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.

The Layer 3 matching is done only when the L3SAM0 or L3DAM0 bit is set.

**Layer4 Address filter 0 register (ETH_MACL4A0R)**

Address offset: 0x0904
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>L4DP0[15:0]</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>L4SP0[15:0]</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 31:16  **L4DP0[15:0]**: Layer 4 Destination Port Number Field
When the L4PEN0 bit is reset and the L4DPM0 bit is set in the **L3 and L4 control 0 register (ETH_MACL3L4C0R)**, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in **L3 and L4 control 0 register (ETH_MACL3L4C0R)**, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

Bits 15:0  **L4SP0[15:0]**: Layer 4 Source Port Number Field
When the L4PEN0 bit is reset and the L4DPM0 bit is set in the **L3 and L4 control 0 register (ETH_MACL3L4C0R)**, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in **L3 and L4 control 0 register (ETH_MACL3L4C0R)**, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.
**Layer3 Address 0 filter 0 register (ETH_MACL3A00R)**

Address offset: 0x0910  
Reset value: 0x0000 0000  
For IPv4 packets, the Layer 3 Address 0 filter 0 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

<table>
<thead>
<tr>
<th>Bit 31-16</th>
<th>Bit 15-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3A00[31:16]</td>
<td>L3A00[15:0]</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>1 0 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

**Bits 31:0 L3A00[31:0]: Layer 3 Address 0 Field**  
- When the L3PEN0 and L3SAM0 bits are set in the **L3 and L4 control 0 register (ETH_MACL3L4C0R)**, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.  
- When the L3PEN0 and L3DAM0 bits are set in the **L3 and L4 control 0 register (ETH_MACL3L4C0R)**, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.  
- When the L3PEN0 bit is reset and the L3SAM0 bit is set in the **L3 and L4 control 0 register (ETH_MACL3L4C0R)**, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

**Layer3 Address 1 filter 0 register (ETH_MACL3A10R)**

Address offset: 0x0914  
Reset value: 0x0000 0000  
For IPv4 packets, the Layer 3 Address 1 filter 0 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

<table>
<thead>
<tr>
<th>Bit 31-16</th>
<th>Bit 15-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3A10[31:16]</td>
<td>L3A10[15:0]</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>1 0 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31-16</th>
<th>Bit 25-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3A10[25:16]</td>
<td>L3A10[15:0]</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>1 0 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>
Layer 3 Address 2 filter 0 register (ETH_MACL3A20R)

Address offset: 0x0918
Reset value: 0x0000 0000

The Layer 3 Address 2 filter 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

Layer 3 Address 3 filter 0 register (ETH_MACL3A30R)

Address offset: 0x091C
Reset value: 0x0000 0000

The Layer 3 Address 3 filter 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.
RM0481 Ethernet (ETH): media access control (MAC) with DMA controller

Bits 31:0  **L3A30[31:0]:** Layer 3 Address 3 Field
When the L3PEN0 and L3SAM0 bits are set in the **L3 and L4 control 0 register (ETH_MACL3L4C0R),** this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.
When the L3PEN0 and L3DAM0 bits are set in the **L3 and L4 control 0 register (ETH_MACL3L4C0R),** this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.
When the L3PEN0 bit is reset in the **L3 and L4 control 0 register (ETH_MACL3L4C0R),** this field is not used.

**L3 and L4 control 1 register (ETH_MACL3L4C1R)**

Address offset: 0x0930
Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 1 of Layer 3 and Layer 4.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3HDBM1[4:0]</td>
<td>L3HSBM1[4:0]</td>
<td>L3DAIM1</td>
<td>L3DAM1</td>
<td>L3SAIM1</td>
<td>L3SAM1</td>
<td>Res.</td>
<td>Res.</td>
<td>Res.</td>
<td>Res.</td>
<td>L4DPM1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:22  Reserved, must be kept at reset value.

Bit 21  **L4DPM1:** Layer 4 Destination Port Match Enable
When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.
This bit is valid and applicable only when the L4DPM1 bit is set high.

Bit 20  **L4DPM1:** Layer 4 Destination Port Match Enable
When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.

Bit 19  **L4SPIM1:** Layer 4 Source Port Inverse Match Enable
When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.
This bit is valid and applicable only when the L4SPIM1 bit is set high.

Bit 18  **L4SPM1:** Layer 4 Source Port Match Enable
When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.

Bit 17  Reserved, must be kept at reset value.
Bit 16 **L4PEN1**: Layer 4 Protocol Enable

When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.

The Layer 4 matching is done only when the L4SPM1 or L4DPM1 bit is set.

Bits 15:11 **L3HDBM1[4:0]**: Layer 3 IP DA higher bits match

Condition: IPv4 packets

This field contains the number of lower bits of IP Destination Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.
1: LSb[0] is masked
2: Two LSbs [1:0] are masked
...
31: All bits except MSb are masked.

Condition: IPv6 packets

Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM1, which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM1[1:0] and L3HSBM1 bits:

0: No bits are masked
1: LSb[0] is masked
2: Two LSbs [1:0] are masked
...
127: All bits except MSb are masked

This field is valid and applicable only when the L3DAM1 or L3SAM1 bit is set.

Bits 10:6 **L3HSBM1[4:0]**: Layer 3 IP SA Higher Bits Match

Condition: IPv4 packets

This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.
1: LSb[0] is masked
2: Two LSbs [1:0] are masked
...
31: All bits except MSb are masked.

Condition: IPv6 packets

This field contains Bits[4:0] of L3HSBM1. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM1 or L3SAM1 bit is set high.

Bit 5 **L3DAIM1**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM1 bit is set high.

Bit 4 **L3DAM1**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

*Note*: When the L3PEN1 bit is set, you should set either this bit or the L3SAM1 bit because either IPv6 DA or SA can be checked for filtering.
Bit 3 **L3SAIM1**: Layer 3 IP SA Inverse Match Enable  
When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching.  
When this bit is reset, the Layer 3 IP Source Address field is enabled for perfect matching.  
This bit is valid and applicable only when the L3SAM1 bit is set.

Bit 2 **L3SAM1**: Layer 3 IP SA Match Enable  
When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.  
**Note:** When the L3PEN01 bit is set, you should set either this bit or the L3DAM1 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **L3PEN1**: Layer 3 Protocol Enable  
When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.  
The Layer 3 matching is done only when the L3SAM1 or L3DAM1 bit is set.

### Layer 4 address filter 1 register (ETH_MACL4A1R)

Address offset: 0x0934  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16 **L4DP1[15:0]**: Layer 4 Destination Port Number Field  
When the L4PEN1 bit is reset and the L4DPM1 bit is set in the L3 and L4 control 1 register (ETH_MACL3L4C1R), this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.  
When the L4PEN1 and L4DPM1 bits are set in L3 and L4 control 1 register (ETH_MACL3L4C1R), this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

Bits 15:0 **L4SP1[15:0]**: Layer 4 Source Port Number Field  
When the L4PEN1 bit is reset and the L4DPM1 bit is set in the L3 and L4 control 1 register (ETH_MACL3L4C1R), this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.  
When the L4PEN1 and L4DPM1 bits are set in L3 and L4 control 1 register (ETH_MACL3L4C1R), this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.
Layer3 address 0 filter 1 Register (ETH_MACL3A01R)

Address offset: 0x0940
Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 filter 1 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3A01[31:16]</td>
<td>rw</td>
<td>Layer 3 Address 0 Field</td>
</tr>
<tr>
<td>L3A01[15:0]</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **L3A01[31:0]**: Layer 3 Address 0 Field
- When the L3PEN1 and L3SAM1 bits are set in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.
- When the L3PEN1 and L3DAM1 bits are set in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.
- When the L3PEN1 bit is reset and the L3SAM1 bit is set in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer3 address 1 filter 1 register (ETH_MACL3A11R)

Address offset: 0x0944
Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 filter 1 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3A11[31:16]</td>
<td>rw</td>
<td>Layer 3 Address 1 Field</td>
</tr>
<tr>
<td>L3A11[15:0]</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **L3A11[31:0]**: Layer 3 Address 1 Field
- When the L3PEN1 and L3SAM1 bits are set in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.
- When the L3PEN1 and L3DAM1 bits are set in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.
- When the L3PEN1 bit is reset and the L3SAM1 bit is set in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.
Bits 31:0 **L3A11[31:0]**: Layer 3 Address 1 Field
When the L3PEN1 and L3SAM1 bits are set in the **L3 and L4 control 1 register** ([ETH_MACL3L4C1R](#)), this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.
When the L3PEN1 and L3DAM1 bits are set in the **L3 and L4 control 1 register** ([ETH_MACL3L4C1R](#)), this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.
When the L3PEN1 bit is reset and the L3SAM1 bit is set in the **L3 and L4 control 1 register** ([ETH_MACL3L4C1R](#)), this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

**Layer3 address 2 filter 1 Register (ETH_MACL3A21R)**
Address offset: 0x0948
Reset value: 0x0000 0000
The Layer 3 Address 2 filter 1 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **L3A21[31:16]**: Layer 3 Address 2 Field
When the L3PEN1 and L3SAM1 bits are set in the **L3 and L4 control 1 register** ([ETH_MACL3L4C1R](#)), this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.
When the L3PEN1 and L3DAM1 bits are set in the **L3 and L4 control 1 register** ([ETH_MACL3L4C1R](#)), this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.
When the L3PEN1 bit is reset in the **L3 and L4 control 1 register** ([ETH_MACL3L4C1R](#)), this field is not used.

**Layer3 address 3 filter 1 register (ETH_MACL3A31R)**
Address offset: 0x94C
Reset value: 0x0000 0000
The Layer 3 Address 3 filter 1 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **L3A31[31:16]**: Layer 3 Address 3 Field
When the L3PEN1 and L3SAM1 bits are set in the **L3 and L4 control 1 register** ([ETH_MACL3L4C1R](#)), this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.
When the L3PEN1 and L3DAM1 bits are set in the **L3 and L4 control 1 register** ([ETH_MACL3L4C1R](#)), this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.
When the L3PEN1 bit is reset in the **L3 and L4 control 1 register** ([ETH_MACL3L4C1R](#)), this field is not used.
Bits 31:0  **L3A31[31:0]: Layer 3 Address 3 Field**  
When the L3PEN1 and L3SAM1 bits are set in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field is not used.

**Timestamp control Register (ETH_MACTSCR)**

Address offset: 0x0B00

Reset value: 0x0000 2000

This register controls the operation of the System Time generator and processing of PTP packets for timestamping in the Receiver.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:29  Reserved, must be kept at reset value.

**Bit 28 AV8021ASMEN: AV 802.1AS Mode Enable**

When this bit is set, the MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing timestamp snapshots, that is, IEEE 802.1AS operating mode.

When PTP offload feature is enabled, for the purpose of PTP offload, the transport specific field in the PTP header is generated and checked based on the value of this bit.

Bits 27:25  Reserved, must be kept at reset value.

**Bit 24 TTXTSSTM: Transmit Timestamp Status Mode**

When this bit is set, the MAC overwrites the earlier transmit timestamp status even if it is not read by the software. The MAC indicates this by setting the TTXTSSMIS bit of the **Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)** register.

When this bit is reset, the MAC ignores the timestamp status of current packet if the timestamp status of previous packet is not read by the software. The MAC indicates this by setting the TTXTSSMIS bit of the **Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)**.

Bits 23:19  Reserved, must be kept at reset value.
Bit 18 **TSENMACADDR**: Enable MAC Address for PTP Packet Filtering  
When this bit is set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP packets when PTP is directly sent over Ethernet.  
When this bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programmed in MAC address registers are considered for processing as indicated below, when PTP is directly sent over Ethernet.  
For normal timestamping operation, MAC address registers 0 to 31 is considered for unicast destination address matching.  
For PTP offload, only MAC address register 0 is considered for unicast destination address matching.

Bits 17:16 **SNAPTYPESEL[1:0]**: Select PTP packets for Taking Snapshots  
These bits, along with Bits 15 and 14, define the set of PTP packet types for which snapshot needs to be taken. The encoding is given in Table 635: Timestamp Snapshot Dependency on ETH_MACTSCR bits.

Bit 15 **TSMSTRENA**: Enable Snapshot for Messages Relevant to Master  
When this bit is set, the snapshot is taken only for the messages that are relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node.

Bit 14 **TSEVNTENA**: Enable Timestamp Snapshot for Event Messages  
When this bit is set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When this bit is reset, the snapshot is taken for all messages except Announce, Management, and Signaling. For more information about the timestamp snapshots, see Table 635: Timestamp Snapshot Dependency on ETH_MACTSCR bits.

Bit 13 **TSIPV4ENA**: Enable Processing of PTP Packets Sent over IPv4-UDP  
When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When this bit is reset, the MAC ignores the PTP transported over IPv4-UDP packets. This bit is set by default.

Bit 12 **TSIPV6ENA**: Enable Processing of PTP Packets Sent over IPv6-UDP  
When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When this bit is clear, the MAC ignores the PTP transported over IPv6-UDP packets.

Bit 11 **TSIPENA**: Enable Processing of PTP over Ethernet Packets  
When this bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When this bit is reset, the MAC ignores the PTP over Ethernet packets.

Bit 10 **TSVER2ENA**: Enable PTP Packet Processing for Version 2 Format  
When this bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When this bit is reset, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in 'PTP Processing and Control'.

Bit 9 **TSCTRLSSR**: Timestamp Digital or Binary Rollover Control  
When this bit is set, the Timestamp Low register rolls over after 0x3B9A_C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When this bit is reset, the rollover value of subsecond register is 0x7FFF_FFFF. The subsecond increment must be programmed correctly depending on the PTP reference clock frequency and the value of this bit.

Bit 8 **TSENALL**: Enable Timestamp for All Packets  
When this bit is set, the timestamp snapshot is enabled for all packets received by the MAC.

Bits 7:6 Reserved, must be kept at reset value.
Bit 5  **TSADDR**: Update Addend Register
When this bit is set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This bit is cleared when the update is complete. This bit should be zero before it is set.

Bit 4  Reserved, must be kept at reset value.

Bit 3  **TSUPDT**: Update Timestamp
When this bit is set, the system time is updated (added or subtracted) with the value specified in System time seconds update register (ETH_MACSTSUR) and System time nanoseconds update register (ETH_MACSTNUR).
This bit should be zero before updating it. This bit is reset when the update is complete in hardware.

Bit 2  **TSINIT**: Initialize Timestamp
When this bit is set, the system time is initialized (overwritten) with the value specified in the System time seconds update register (ETH_MACSTSUR) and System time nanoseconds update register (ETH_MACSTNUR).
This bit should be zero before it is updated. This bit is reset when the initialization is complete.

Bit 1  **TSCFUPDT**: Fine or Coarse Timestamp Update
When this bit is set, the Fine method is used to update system timestamp. When this bit is reset, Coarse method is used to update the system timestamp.

Bit 0  **TSENA**: Enable Timestamp
When this bit is set, the timestamp is added for Transmit and Receive packets. When disabled, timestamp is not added for transmit and receive packets and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode.
On the Receive side, the MAC processes the 1588 packets only if this bit is set.

**Subsecond increment register (ETH_MACSSIR)**
Address offset: 0x0B04
Reset value: 0x0000 0000

In Coarse Update mode (bit TSCFUPDT in Timestamp control Register (ETH_MACTSCR)), the value in this register is added to the system time every clock cycle of clk_ptp_ref_i. In Fine Update mode, the value in this register is added to the system time whenever the Accumulator gets an overflow.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>SSINC[7:0]</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:24  Reserved, must be kept at reset value.

Bits 23:16  **SSINC[7:0]**: Subsecond Increment Value

The value programmed in this field is accumulated every clock cycle (of clk_ptp_i) with the contents of the subsecond register. For example, when the PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time Nanoseconds register has an accuracy of 1 ns [TSCTRLSSR bit is set in *Timestamp control Register (ETH_MAC_TSCR)*]. When TSCTRLSSR is cleared, the Nanoseconds register has a resolution of ~0.465 ns. In this case, you should program a value of 43 (0x2B) which is derived by 20 ns/0.465.

Bits 15:0  Reserved, must be kept at reset value.
**System time seconds register (ETH_MACSTSR)**

Address offset: 0xB08
Reset value: 0x0000 0000

The System Time Seconds register, along with System Time Nanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to CSR clock).

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r  r  r  r  r  r  r  r  r  r  r  r  r  r  r  r</td>
</tr>
<tr>
<td>TSS[31:16]</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r  r  r  r  r  r  r  r  r  r  r  r  r  r  r  r</td>
</tr>
<tr>
<td>TSS[15:0]</td>
</tr>
</tbody>
</table>

Bits 31:0 **TSS[31:0]:** Timestamp Second

The value in this field indicates the current value in seconds of the System Time maintained by the MAC.

**System time nanoseconds register (ETH_MACSTNR)**

Address offset: 0xB0C
Reset value: 0x0000 0000

The System Time Nanoseconds register, along with System Time Seconds register, indicates the current value of the system time maintained by the MAC.

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r  r  r  r  r  r  r  r  r  r  r  r  r  r  r  r</td>
</tr>
<tr>
<td>TSSS[30:16]</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r  r  r  r  r  r  r  r  r  r  r  r  r  r  r  r</td>
</tr>
<tr>
<td>TSSS[15:0]</td>
</tr>
</tbody>
</table>

Bit 31 **Reserved, must be kept at reset value.**

Bits 30:0 **TSSS[30:0]:** Timestamp subseconds

The value in this field has the subsecond representation of time, with an accuracy of 0.46 ns. When TSCTRLSSR is set in **Timestamp control Register (ETH_MACTSCR)**, each bit represents 1 ns. The maximum value is 0x3B9A_C9FF after which it rolls-over to zero.
System time seconds update register (ETH_MACSTSUR)

Address offset: 0x0B10
Reset value: 0x0000 0000

The System Time Seconds Update register, along with the System Time Nanoseconds update register, initializes or updates the system time maintained by the MAC. You must write both registers before setting the TSINIT or TSUPDT bits in Timestamp control Register (ETH_MACTSCR).

<table>
<thead>
<tr>
<th>TSS[31:16]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TSS[15:0]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
</table>

Bits 31:0 TSS[31:0]: Timestamp Seconds
The value in this field is the seconds part of the update.
When ADDSUB is reset, this field must be programmed with the seconds part of the update value.
When ADDSUB is set, this field must be programmed with the complement of the seconds part of the update value.
For example, to subtract 2.000000001 seconds from the system time, the TSS field in the ETH_MACSTSUR register must be 0xFFFF_FFFE (that is, 2^32 – 2).

Caution: When the ADDSUB bit is set, TSS[30:0] field cannot be set to 0 in System time nanoseconds update register (ETH_MACSTNUR). The TSSS bitfield must be programmed to 0x7FFF FFFF (resulting in −0.46 ns) even if 0 ns must be subtracted.
For example, to subtract 2.000000000 seconds from the system time, the TSS field in the System time seconds update register (ETH_MACSTSUR) must be 0xFFFF FFFF (that is, 2^32 – 1) and the System time nanoseconds update register (ETH_MACSTNUR) must be 0xFFFF FFFF (ADDSUB = 1 and TSSS[30:0] field = 0x7FFF_FFFF)

System time nanoseconds update register (ETH_MACSTNUR)

Address offset: 0x0B14
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>ADD SUB</th>
<th>TSS[30:16]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TSSS[15:0]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
</table>
Ethernet (ETH): media access control (MAC) with DMA controller

Bit 31 ADDSUB: Add or Subtract Time
   When this bit is set, the time value is subtracted with the contents of the update register.
   When this bit is reset, the time value is added with the contents of the update register.

Bits 30:0 TSSS[30:0]: Timestamp subseconds
   The value in this field is the subseconds part of the update.
      ADDSUB is 1: This field must be programmed with the complement of the subseconds part of the update value as described.
      ADDSUB is 0: This field must be programmed with the subseconds part of the update value, with an accuracy based on the TSCTRLSSR bit of the Timestamp control Register (ETH_MACTSCR).
   TSCTRLSSR field in the Timestamp control Register (ETH_MACTSCR) is 1:
      - The programmed value must be 10^9 – <subsecond value>.
      - Each bit represents 1 ns and the programmed value should not exceed 0xB9A_C9FF.
   TSCTRLSSR field in the Timestamp control Register (ETH_MACTSCR) is 0:
      - The programmed value must be 2^31 - <subsecond_value>.
      - Each bit represents an accuracy of 0.46 ns.
For example, to subtract 2.000000001 seconds from the system time, then the TSSS field in the ETH_MACSTNUR register must be 0x7FFF_FFFF (that is, 2^31 – 1), when TSCTRLSSR bit in Timestamp control Register (ETH_MACTSCR) is reset and 0xB9A_C9FF (that is, 10^9 –1), when TSCTRLSSR bit in Timestamp control Register (ETH_MACTSCR) is set.

Caution: When the ADDSUB bit is set, TSSS[30:0] field cannot be set to 0. The TSSS bitfield must be programmed to 0x7FFF_FFFF (resulting in –0.46 ns) even if 0 ns must be subtracted.
For example, to subtract 2.000000000 seconds from the system time, System time nanoseconds update register (ETH_MACSTNUR) must be 0xFFFF_FFFF (ADDSUB = 1 and TSSS[30:0] = 0) and the TSS field in the System time seconds update register (ETH_MACSTSUR) must be 0xFFFF_FFFE (that is, 2^32 – 1).

Timestamp addend register (ETH_MACTSAR)
Address offset: 0x0B18
Reset value: 0x0000 0000
This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in the Timestamp control Register (ETH_MACTSCR)). The content of this register is added to a 32-bit accumulator in every clock cycle of clk_ptp_ref_i and the system time is updated whenever the accumulator overflows.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31:16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

Bits 31:0 TSAR[31:0]: Timestamp Addend Register
This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.
**Timestamp status register (ETH_MACTSSR)**

Address offset: 0x0B20

Reset value: 0x0000 0000

All bits except Bits[27:25] gets cleared when the application reads this register.

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Bits 31:30  Reserved, must be kept at reset value.

Bits 29:25  **ATSSN[4:0]**: Number of Auxiliary Timestamp Snapshots

This field indicates the number of Snapshots available in the FIFO. A value equal to the depth of FIFO (4) indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 00000) when the Auxiliary snapshot FIFO clear bit is set.

Bit 24  **ATSS TM**: Auxiliary Timestamp Snapshot Trigger Missed

This bit is set when the Auxiliary timestamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot is not stored in the FIFO.

Bits 23:20  Reserved, must be kept at reset value.

Bits 19:16  **ATSSSTN[3:0]**: Auxiliary Timestamp Snapshot Trigger Identifier

These bits identify the Auxiliary trigger inputs for which the timestamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. These bits are applicable only if the number of Auxiliary snapshots is more than one. One bit is assigned for each trigger as shown in the following list:

- Bit 16: Auxiliary trigger 0
- Bit 17: Auxiliary trigger 1
- Bit 18: Auxiliary trigger 2
- Bit 19: Auxiliary trigger 3

The software can read this register to find the triggers that are set when the timestamp is taken.

Bit 15  **TXTSSIS**: Tx Timestamp Status Interrupt Status

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the Tx timestamp status nanoseconds register (ETH_MACTXTSSNR) and Tx timestamp status seconds register (ETH_MACTXTSSSR).

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the Tx timestamp status nanoseconds register (ETH_MACTXTSSNR) and Tx timestamp status seconds register (ETH_MACTXTSSSR), for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the **Tx timestamp status seconds register (ETH_MACTXTSSSR)** is read (or written to 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set).

Bits 14:4  Reserved, must be kept at reset value.
Bit 3 **TSTRGTE0**: Timestamp Target Time Error
This bit is set when the latest target time programmed in the ETH_MACPPSTTSR and ETH_MACPPSTTSNR elapses (see PPS target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTTNR)). This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set).

Bit 2 **AUXSTRIG**: Auxiliary Timestamp Trigger Snapshot
This bit is set high when the auxiliary snapshot is written to the FIFO. This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set).

Bit 1 **TSTARGT0**: Timestamp Target Time Reached
When set, this bit indicates that the value of system time is greater than or equal to the value specified in the ETH_MACPPSTTSR and ETH_MACPPSTTSNR registers (see PPS target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTTNR)). This bit is cleared when the application reads this bit (or writes of 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set).

Bit 0 **TSSOVF**: Timestamp Seconds Overflow
When this bit is set, it indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 32'hFFFF_FFFF. This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set).

**Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)**
Address offset: 0x0B30
Reset value: 0x0000 0000
This register contains the nanosecond part of timestamp captured for Transmit packets when Tx status is disabled.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXTSS MIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| 15:0 | |
|------| |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

Bit 31 **TXTSSMIS**: Transmit Timestamp Status Missed
When this bit is set, it indicates one of the following:
- The timestamp of the current packet is ignored if TXTSSTSM bit of the Timestamp control Register (ETH_MACTSCR) is reset
- The timestamp of the previous packet is overwritten with timestamp of the current packet if TXTSSTSM bit of the Timestamp control Register (ETH_MACTSCR) is set.

Bits 30:0 **TXTSSL[30:0]**: Transmit Timestamp Status Low
This field contains the 31 bits of the Nanoseconds field of the Transmit packet's captured timestamp.
**Tx timestamp status seconds register (ETH_MACTXTSSSR)**

Address offset: 0x0B34  
Reset value: 0x0000 0000  
The register contains the higher 32 bits of the timestamp (in seconds) captured when a PTP packet is transmitted.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXTSSH[31:16]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 **TXTSSH[31:0]**: Transmit Timestamp Status High  
This field contains the lower 32 bits of the Seconds field of Transmit packet's captured timestamp.

**Auxiliary control register (ETH_MACACR)**

Address offset: 0x0B40  
Reset value: 0x0000 0000  
The Auxiliary Timestamp Control register controls the Auxiliary Timestamp snapshot.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
<td>Aux</td>
</tr>
</tbody>
</table>

Bits 31:8  
**Reserved, must be kept at reset value.**

**Bit 7 **ATSEN3: Auxiliary Snapshot 3 Enable  
– This bit controls the capturing of Auxiliary Snapshot Trigger 3. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg3 input is enabled. When this bit is reset, the events on this input are ignored.

**Bit 6 **ATSEN2: Auxiliary Snapshot 2 Enable  
– This bit controls the capturing of Auxiliary Snapshot Trigger 2. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg2 input is enabled. When this bit is reset, the events on this input are ignored.

**Bit 5 **ATSEN1: Auxiliary Snapshot 1 Enable  
– This bit controls the capturing of Auxiliary Snapshot Trigger 1. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg1 input is enabled. When this bit is reset, the events on this input are ignored.
Bit 4 **ATSEN0**: Auxiliary Snapshot 0 Enable
This bit controls the capturing of Auxiliary Snapshot Trigger 0. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg0 input is enabled. When this bit is reset, the events on this input are ignored.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ATSFC**: Auxiliary Snapshot FIFO Clear
When set, this bit resets the pointers of the Auxiliary Snapshot FIFO. This bit is cleared when the pointers are reset and the FIFO is empty. When this bit is high, the auxiliary snapshots are stored in the FIFO.

**Auxiliary timestamp nanoseconds register (ETH_MACATSNR)**

Address offset: 0x0B48
Reset value: 0x0000 0000

The *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)*, along with *Auxiliary timestamp seconds register (ETH_MACATSSR)*, gives the 64-bit timestamp stored as auxiliary snapshot. These two registers form the read port of a 64-bit wide FIFO with a depth of 4 words.

You can store multiple snapshots in this FIFO. Bits[29:25] in *Timestamp status register (ETH_MACTSSR)* indicate the fill-level of the FIFO. The top of the FIFO is removed only when *Auxiliary timestamp seconds register (ETH_MACATSSR)* is read.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**AUXTSLO[30:16]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**AUXTSLO[15:0]**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **AUXTSLO[30:0]**: Auxiliary Timestamp
Contains the lower 31 bits (nanoseconds field) of the auxiliary timestamp.

**Auxiliary timestamp seconds register (ETH_MACATSSR)**

Address offset: 0x0B4C
Reset value: 0x0000 0000

The Auxiliary Timestamp Seconds register contains the lower 32 bits of the Seconds field of the auxiliary timestamp register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**AUCTSHI[31:16]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**AUCTSHI[15:0]**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2920/3119
Bits 31:0  **AUXTSHI[31:0]**: Auxiliary Timestamp  
Contains the lower 32 bits of the Seconds field of the auxiliary timestamp.

**Timestamp Ingress asymmetric correction register (ETH_MACTSIACR)**  
Address offset: 0x0B50  
Reset value: 0x0000 0000  
The MAC Timestamp Ingress Asymmetry Correction register contains the Ingress Asymmetry Correction value to be used while updating correction field in PDelay_Resp PTP messages.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>OSTIAC[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 31:0  **OSTIAC[31:0]**: One-Step Timestamp Ingress Asymmetry Correction  
This field contains the ingress path asymmetry value to be added to correctionField of Pdelay_Resp PTP packet. The programmed value should be in units of nanoseconds and multiplied by 2^16. For example, 2.5 ns is represented as 0x00028000. The value can also be negative, which is represented in 2's complement form with bit 31 representing the sign bit.

**Timestamp Egress asymmetric correction register (ETH_MACTSEACR)**  
Address offset: 0x0B54  
Reset value: 0x0000 0000  
The MAC Timestamp Egress Asymmetry Correction register contains the Egress Asymmetry Correction value to be used while updating the correction field in PDelay_Req PTP messages.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>OSTEAC[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 31:0  **OSTEAC[31:0]**: One-Step Timestamp Egress Asymmetry Correction  
This field contains the egress path asymmetry value to be subtracted from correctionField of Pdelay_Resp PTP packet. The programmed value must be the negated value in units of nanoseconds multiplied by 2^16. For example, if the required correction is +2.5 ns, the programmed value must be 0xFFFF_8000, which is the 2's complement of 0x0002_8000(2.5 * 2^16). Similarly, if the required correction is -3.3 ns, the programmed value is 0x0003_4CCC (3.3 * 2^16).
Timestamp Ingress correction nanosecond register (ETH_MACTSICNR)

Address offset: 0x0B58

Reset value: 0x0000 0000

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the ingress path.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **TSIC[31:0]**: Timestamp Ingress Correction

This field contains the ingress path correction value as defined by the Ingress Correction expression.

Timestamp Egress correction nanosecond register (ETH_MACTSECNR)

Address offset: 0x0B5C

Reset value: 0x0000 0000

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the egress path.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **TSEC[31:0]**: Timestamp Egress Correction

This field contains the nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

PPS control register (ETH_MACPPSCR)

Address offset: 0x0B70

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TRGMODESEL0[1:0]</td>
<td>PPSEN</td>
<td>PPSCTRL[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:7 Reserved, must be kept at reset value.

Bits 6:5 TRGTMODSEL[1:0]: Target Time Register Mode for PPS Output
This field indicates the Target Time registers (PPS target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTTNR)) mode for PPS output signal:
00: Target Time registers are programmed only for generating the interrupt event.
01: Reserved, must not be used
10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.
11: Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 4 PPSEN0: Flexible PPS Output Mode Enable
When this bit is set, PPSCTRL[3:0] function as PPSCMD[3:0]. When this bit is reset, PPSCTRL[3:0] function as PPSCTRL (Fixed PPS mode).

Bits 3:0 PPSCTRL[3:0]: PPS Output Frequency Control
This field controls the frequency of the PPS output (eth_ptp_pps_out) signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies:
0001: The binary rollover is 2 Hz, and the digital rollover is 1 Hz.
0010: The binary rollover is 4 Hz, and the digital rollover is 2 Hz.
0011: The binary rollover is 8 Hz, and the digital rollover is 4 Hz.
0100: The binary rollover is 16 Hz, and the digital rollover is 8 Hz.

.. 1111: The binary rollover is 32.768 KHz and the digital rollover is 16.384 KHz.

Note: In the binary rollover mode, the PPS output (eth_ptp_pps_out) has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example:
- When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms
- When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of
  One clock of 50 percent duty cycle and 537 ms period
  Second clock of 463 ms period (268 ms low and 195 ms high)
- When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of
  Three clocks of 50 percent duty cycle and 268 ms period
  Fourth clock of 195 ms period (134 ms low and 61 ms high)

This behavior is because of the non-linear toggling of bits in the digital rollover mode in the ETH_MACSTNR register.
**PPS control register [alternate] (ETH_MACPPSCR)**

Address offset: 0x0B70  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:5 **TRGTMODSEL0[1:0]**: Target Time Register Mode for PPS Output

- This field indicates the Target Time registers ([**PPS target time seconds register (ETH_MACPPSTTSR)**](#) and **PPS target time nanoseconds register (ETH_MACPPSTTNR)**) mode for PPS output signal:
  - 00: Target Time registers are programmed only for generating the interrupt event.
  - 01: Reserved, must not be used
  - 10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.
  - 11: Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 4 **PPSEN0**: Flexible PPS Output Mode Enable

When this bit is set, Bits[3:0] function as PPSCMD[3:0]. When this bit is reset, Bits[3:0] function as PPSCTRL(Fixed PPS mode).
Bits 3:0 **PPSCMD[3:0]**: Flexible PPS Output (eth_ptp_pps_out) Control

Programming these bits with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, these bits get cleared automatically. The software should ensure that these bits are programmed only when they are 'all zero'. The following list describes the values of PPSCMD[3:0]:

- **0000**: No Command
- **0001**: START Single Pulse.
  This command generates single pulse rising at the start point defined in Target Time Registers (register 455 and 456) and of a duration defined in the PPS Width Register.
- **0010**: START Pulse Train.
  This command generates the train of pulses rising at the start point defined in the Target Time Registers and of a duration defined in the PPS Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by the 'Stop Pulse train at time' or 'Stop Pulse Train immediately' commands.
- **0011**: Cancel START.
  This command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programmed start time.
- **0100**: STOP Pulse Train at time.
  This command stops the train of pulses initiated by the START Pulse Train command (PPSCMD[3:0] = 0010) after the time programmed in the Target Time registers elapses.
- **0101**: STOP Pulse Train immediately.
  This command immediately stops the train of pulses initiated by the START Pulse Train command (PPSCMD[3:0] = 0010).
- **0110**: Cancel STOP Pulse train.
  This command cancels the STOP pulse train at time command if the programmed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command.
- **0111 to 1111**: Reserved, must not be used

**PPS target time seconds register (ETH_MACPPSTTSR)**

Address offset: 0x0B80
Reset value: 0x0000 0000

The PPS output Target Time Seconds register, along with **PPS target time nanoseconds register (ETH_MACPPSTTNR)**, is used to schedule an interrupt event (Bit TSSOVF of Timestamp status register (ETH_MACTSSR)) when the system time exceeds the value programmed in these registers.

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
```

**ST**

RM0481 Rev 1  2925/3119
Bits 31:0 **TSTRH0[31:0]**: PPS Target Time Seconds Register
This field stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on Target Time mode selected for the corresponding PPS output in the **PPS control register (ETH_MACPPSCR)**.

**PPS target time nanoseconds register (ETH_MACPPSTTNR)**
Address offset: 0x0B84
Reset value: 0x0000 0000
The PPS Target Time Nanoseconds register is present only when more than one Flexible PPS output is selected.

<table>
<thead>
<tr>
<th>TRGTBUSY0</th>
<th>TTSL0[30:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

| TTSL0[15:0] |
| rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw |

Bit 31 **TRGTBUSY0**: PPS Target Time Register Busy
The MAC sets this bit when the PPSCMD0 field in the **PPS control register (ETH_MACPPSCR)** is programmed to 010 or 011. Programming the PPSCMD0 field to 010 or 011 instructs the MAC to synchronize the Target Time Registers to the PTP clock domain. The MAC clears this bit after synchronizing the Target Time Registers with the PTP clock domain. The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted.

**PPS interval register (ETH_MACPPPSIR)**
Address offset: 0x0B88
Reset value: 0x0000 0000
The PPS Interval register contains the number of units of subsecond increment value between the rising edges of PPS output.
RM0481 Ethernet (ETH): media access control (MAC) with DMA controller

PPS INT0[31:16]: PPS Output Signal Interval

These bits store the interval between the rising edges of PPS signal output. The interval is stored in terms of number of units of subsecond increment value. You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and desired interval between the rising edges of PPS signal output is 100 ns (that is, 5 units of subsecond increment value), you should program value 4 (5-1) in this register.

**PPS width register (ETH_MACPPSWR)**

Address offset: 0x0B8C
Reset value: 0x0000 0000
The PPS Width register contains the number of units of subsecond increment value between the rising and corresponding falling edges of PPS output.

Bits 31:0 PPS WIDTH0[31:0]: PPS Output Signal Width

These bits store the width between the rising edge and corresponding falling edge of PPS signal output. The width is stored in terms of number of units of subsecond increment value. You need to program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20 ns), and width between the rising and corresponding falling edges of PPS signal output is 80 ns (that is, four units of subsecond increment value), you should program value 3 (4-1) in this register.

**Note:** The value programmed in this register must be lesser than the value programmed in PPS interval register (ETH_MACPPSIR).
PTP Offload control register (ETH_MACPOCR)

Address offset: 0x0BC0
Reset value: 0x0000 0000

This register controls the PTP Offload Engine operation.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DN[7:0]</th>
<th>DRRDIS</th>
<th>APDREQTRIG</th>
<th>ASYNCTRIG</th>
<th>APDREQEN</th>
<th>ASYNCE</th>
<th>PTOEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 DN[7:0]: Domain Number
This field indicates the domain Number in which the PTP node is operating.

Bit 7 Reserved, must be kept at reset value.

Bit 6 DRRDIS: Disable PTO Delay Request/Response response generation
When this bit is set, the Delay Request and Delay response are not generated for received
SYNC and Delay request packet respectively, as required by the programmed mode.

Bit 5 APDREQTRIG: Automatic PTP Pdelay_Req message Trigger
When this bit is set, one PTP Pdelay_Req message is transmitted. This bit is automatically
cleared after the PTP Pdelay_Req message is transmitted. The application should set the
APDREQEN bit for this operation.

Bit 4 ASYNCTRIG: Automatic PTP SYNC message Trigger
When this bit is set, one PTP SYNC message is transmitted. This bit is automatically cleared
after the PTP SYNC message is transmitted. The application should set the ASYNCE bit for
this operation.

Bit 3 Reserved, must be kept at reset value.

Bit 2 APDREQEN: Automatic PTP Pdelay_Req message Enable
When this bit is set, PTP Pdelay_Req message is generated periodically based on interval
programmed or trigger from application, when the MAC is programmed to be in Peer-to-Peer
Transparent mode.

Bit 1 ASYNCE: Automatic PTP SYNC message Enable
When this bit is set, PTP SYNC message is generated periodically based on interval
programmed or trigger from application, when the MAC is programmed to be in Clock Master
mode.

Bit 0 PTOEN: PTP Offload Enable
When this bit is set, the PTP Offload feature is enabled.

PTP Source Port Identity 0 Register (ETH_MACSPI0R)

Address offset: 0x0BC4
Reset value: 0x0000 0000

This register contains Bits[31:0] of the 80-bit Source Port Identity of the PTP node.
RM0481  Ethernet (ETH): media access control (MAC) with DMA controller

**PTP Source port identity 1 register (ETH_MACSPI1R)**

Address offset: 0x0BC8  
Reset value: 0x0000 0000  
This register contains Bits[63:32] of the 80-bit Source Port Identity of the PTP node.

**PTP Source port identity 2 register (ETH_MACSPI2R)**

Address offset: 0x0BCC  
Reset value: 0x0000 0000  
This register contains Bits[79:64] of the 80-bit Source Port Identity of the PTP node.

Bits 31:0  **SPI0[31:0]**: Source Port Identity 0  
This field indicates bits [31:0] of sourcePortIdentity of PTP node.

Bits 31:0  **SPI1[31:0]**: Source Port Identity 1  
This field indicates bits [63:32] of sourcePortIdentity of PTP node.

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **SPI2[15:0]**: Source Port Identity 2  
This field indicates bits [79:64] of sourcePortIdentity of PTP node.
Log message interval register (ETH_MACLMIR)

Address offset: 0x0BD0
Reset value: 0x0000 0000

This register contains the periodic intervals for automatic PTP packet generation.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMPDRI[7:0]</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:24 **LMPDRI[7:0]**: Log Min Pdelay_Req Interval
This field indicates logMinDelayReqInterval of PTP node. This is used to schedule the periodic Pdelay request packet transmission. Allowed values are -15 to 15. Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.

Bits 23:11 Reserved, must be kept at reset value.

Bits 10:8 **DRSYNCR[2:0]**:
Delay_Req to SYNC Ratio
In Slave mode, it is used for controlling frequency of Delay_Req messages transmitted.
0: DelayReq generated for every received SYNC
1: DelayReq generated every alternate reception of SYNC
2: for every 4 SYNC messages
3: for every 8 SYNC messages
4: for every 16 SYNC messages
5: for every 32 SYNC messages
Others: Reserved, must not be used
The master sends this information (logMinDelayReqInterval) in the DelayResp PTP messages to the slave. The reception processes this value from the received DelayResp messages and updates this field accordingly. In the Slave mode, the host must not write/update this register unless it has to override the received value. In Master mode, the sum of this field and logSyncInterval (LSI) field is provided in the logMinDelayReqInterval field of the generated multicast Delay_Resp PTP message.

Bits 7:0 **LSI[7:0]**:
Log Sync Interval
This field indicates the periodicity of the automatically generated SYNC message when the PTP node is Master. Allowed values are -15 to 15. Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.
# Ethernet MAC register map and reset values

Table 682. Ethernet MAC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>ETH_MACCR</td>
<td>ARPEN, SARCC2-0, IPC, IPG[2:0], GPSLE, SAR, CSR, ACS, VPD, JD, JE, FES, DM, LM, ECRSFD, DO, DCRS, DR, BL[1:0], DC, PRELEN[1:0]</td>
</tr>
<tr>
<td>0x0004</td>
<td>ETH_MACECR</td>
<td>EPGEN, EPGEN, USP, DPM, IPFE, VTEE, HT31T[31:0], GPSL[13:0]</td>
</tr>
<tr>
<td>0x0008</td>
<td>ETH_MACPFR</td>
<td>RA, DMTU, IPFE, VTEE, HT63T[32:10], EIPGT[4:0], EIPGEN</td>
</tr>
<tr>
<td>0x000C</td>
<td>ETH_MACWTR</td>
<td>HT63T[31:0], HT31T[31:0], HT63T[32:10]</td>
</tr>
<tr>
<td>0x0010</td>
<td>ETH_MACT0R</td>
<td>HT31T[31:0]</td>
</tr>
<tr>
<td>0x0014</td>
<td>ETH_MACT1R</td>
<td>HT63T[32:10]</td>
</tr>
<tr>
<td>0x0018 -</td>
<td>ETH_MACVTR</td>
<td>VL[15:0]</td>
</tr>
<tr>
<td>0x0054</td>
<td>ETH_MACVHTR</td>
<td>VLHT[15:0]</td>
</tr>
<tr>
<td>0x005C</td>
<td>ETH_MACVIR</td>
<td>VLT[15:0]</td>
</tr>
</tbody>
</table>
### Table 682. Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0064</td>
<td>ETH_MACIVIR</td>
<td></td>
<td>0x0068</td>
<td>ETH_MACIVIR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x006C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0070</td>
<td>ETH_MACQTXFCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0074</td>
<td>ETH_MACQTXFCR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x007C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0090</td>
<td>ETH_MACRXFCR</td>
<td></td>
<td>0x0094</td>
<td>ETH_MACRXFCR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x009C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00B0</td>
<td>ETH_MACISR</td>
<td></td>
<td>0x00B4</td>
<td>ETH_MACISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00B8</td>
<td>ETH_MACISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00BC</td>
<td>ETH_MACISR</td>
<td></td>
</tr>
<tr>
<td>0x00C0</td>
<td>ETH_MACPCSR</td>
<td></td>
<td>0x00C4</td>
<td>ETH_MACPCSR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00C8</td>
<td>ETH_MACPCSR</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0064</td>
<td>ETH_MACIVIR</td>
<td></td>
<td>0x0068</td>
<td>ETH_MACIVIR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x006C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0070</td>
<td>ETH_MACQTXFCR</td>
<td></td>
<td>0x0074</td>
<td>ETH_MACQTXFCR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x007C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0090</td>
<td>ETH_MACRXFCR</td>
<td></td>
<td>0x0094</td>
<td>ETH_MACRXFCR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x009C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00B0</td>
<td>ETH_MACISR</td>
<td></td>
<td>0x00B4</td>
<td>ETH_MACISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00B8</td>
<td>ETH_MACISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00BC</td>
<td>ETH_MACISR</td>
<td></td>
</tr>
<tr>
<td>0x00C0</td>
<td>ETH_MACPCSR</td>
<td></td>
<td>0x00C4</td>
<td>ETH_MACPCSR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00C8</td>
<td>ETH_MACPCSR</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0064</td>
<td>ETH_MACIVIR</td>
<td></td>
<td>0x0068</td>
<td>ETH_MACIVIR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x006C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0070</td>
<td>ETH_MACQTXFCR</td>
<td></td>
<td>0x0074</td>
<td>ETH_MACQTXFCR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x007C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0090</td>
<td>ETH_MACRXFCR</td>
<td></td>
<td>0x0094</td>
<td>ETH_MACRXFCR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x009C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00B0</td>
<td>ETH_MACISR</td>
<td></td>
<td>0x00B4</td>
<td>ETH_MACISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00B8</td>
<td>ETH_MACISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00BC</td>
<td>ETH_MACISR</td>
<td></td>
</tr>
<tr>
<td>0x00C0</td>
<td>ETH_MACPCSR</td>
<td></td>
<td>0x00C4</td>
<td>ETH_MACPCSR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00C8</td>
<td>ETH_MACPCSR</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0064</td>
<td>ETH_MACIVIR</td>
<td></td>
<td>0x0068</td>
<td>ETH_MACIVIR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x006C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0070</td>
<td>ETH_MACQTXFCR</td>
<td></td>
<td>0x0074</td>
<td>ETH_MACQTXFCR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x007C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0090</td>
<td>ETH_MACRXFCR</td>
<td></td>
<td>0x0094</td>
<td>ETH_MACRXFCR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x009C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00B0</td>
<td>ETH_MACISR</td>
<td></td>
<td>0x00B4</td>
<td>ETH_MACISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00B8</td>
<td>ETH_MACISR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00BC</td>
<td>ETH_MACISR</td>
<td></td>
</tr>
<tr>
<td>0x00C0</td>
<td>ETH_MACPCSR</td>
<td></td>
<td>0x00C4</td>
<td>ETH_MACPCSR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00C8</td>
<td>ETH_MACPCSR</td>
<td></td>
</tr>
</tbody>
</table>
Table 682. Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register description</th>
<th>Offset</th>
<th>Register name</th>
<th>Register description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00D0</td>
<td>ETH_MACLCSR</td>
<td>reset value</td>
<td>0x00D4</td>
<td>ETH_MACLTCR</td>
<td>LST[9:0] TWT[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0x00D8</td>
<td>ETH_MACLETR</td>
<td>LPITCSE LPITE LPITXA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0x00DC</td>
<td>ETH_MAC1USTCR</td>
<td>TIC_1US_CNTR[11:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserved</td>
<td>0x00E0-0x010C</td>
<td>ETH_MACHWF0R</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0x0110</td>
<td>ETH_MACVR</td>
<td>USERVER[7:0] SNPSVER[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0x0114</td>
<td>ETH_MACDR</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0x0118</td>
<td>ETH_MACHWF1R</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0x011C</td>
<td>ETH_MACHWF0R</td>
<td>ACTPHYSEL[2:0] SAVLANS TSSTSEL[1:0] MACADDRSEL[4:0] ADDMACADDSEL[4:0] RXCOESL TXCOESL EEESEL TSSEL ARPOFFSEL MACSEL MINKSEL SMASEL PCSEL VLASH CMASEL GMICSEL MUSSEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reset value</td>
<td>0x0120</td>
<td>ETH_MACHWF1R</td>
<td>L3_4NUM[3:0] HAHTBL[8:0] POLUS RGSEL DSPGEM SPHEN DGEM ADDR6[4:0] ADTHWORD PTOEN Osten TXFIFOIZE[4:0] RXFIFOIZE[4:0]</td>
</tr>
</tbody>
</table>

Reset values are shown in the table.
Table 682. Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset: 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0124</td>
<td>ETH_MACHWF2R</td>
<td>Reset value: 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0128</td>
<td>ETH_MACHWF3R</td>
<td>Reset value: 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x012C - 0x01FC</td>
<td>Reserved</td>
<td>Reset value: 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0200</td>
<td>ETH_MACMDIOAR</td>
<td>Reset value: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0204</td>
<td>ETH_MACMDIODR</td>
<td>Reset value: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0208 - 0x020C</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x0210</td>
<td>ETH_MACPAR</td>
<td>Reset value: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0214 - 0x022C</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x0230</td>
<td>ETH_MACCSR8W</td>
<td>Reset value: 0</td>
</tr>
<tr>
<td>0x0234 - 0x02FC</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x0300</td>
<td>ETH_MACA0HR</td>
<td>Reset value: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0x0304</td>
<td>ETH_MACA0LR</td>
<td>Reset value: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>
### Table 682. Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0308</td>
<td>ETH_MACA1HR</td>
<td>0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
<td>0x03C</td>
<td>ETH_MACA1LR</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0x0310</td>
<td>ETH_MACA2HR</td>
<td>0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
<td>0x0314</td>
<td>ETH_MACA2LR</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0x0318</td>
<td>ETH_MACA3HR</td>
<td>0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
<td>0x031C</td>
<td>ETH_MACA3LR</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0x0320-</td>
<td>ETH_MMC_</td>
<td>Reserved</td>
<td>0x0700</td>
<td>ETH_MMC_</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0704</td>
<td>ETH_MMC_RX_</td>
<td>0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
<td>0x0708</td>
<td>ETH_MMC_TX_</td>
<td>0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>CONTROL</td>
<td></td>
<td></td>
<td>INTERRUPT</td>
<td></td>
</tr>
</tbody>
</table>
### Table 682. Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x070C</td>
<td>ETH_MMC_RX_INTERRUPT_MASK</td>
<td></td>
<td>0x0710</td>
<td>ETH_MMC_TX_INTERRUPT_MASK</td>
<td></td>
<td>0x0714 - 0x0748</td>
<td>Reserved</td>
<td>0x074C</td>
</tr>
</tbody>
</table>
Table 682. Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x07C8 - 0x07E8</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x07EC</td>
<td>ETH_TX_LPI_USEC_CNTR</td>
<td>TXLPIUSC[31:0]</td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x07F0</td>
<td>ETH_TX_LPI_TRAN_CNTR</td>
<td>TXLPITRC[31:0]</td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x07F4</td>
<td>ETH_RX_LPI_USEC_CNTR</td>
<td>RXLPIUSC[31:0]</td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x07F8</td>
<td>ETH_RX_LPI_TRAN_CNTR</td>
<td>RXLPITRC[31:0]</td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x07FC - 0x08FC</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0900</td>
<td>ETH_MACL3L4C0R</td>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0904</td>
<td>ETH_MACL4A0R</td>
<td>L4DP0[15:0]</td>
<td>L4SP0[15:0]</td>
</tr>
<tr>
<td>0x0908 - 0x090C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0910</td>
<td>ETH_MACL3A0R</td>
<td>L3A00[31:0]</td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0914</td>
<td>ETH_MACL3A1R</td>
<td>L3A10[31:0]</td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0918</td>
<td>ETH_MACL3A2R</td>
<td>L3A20[31:0]</td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x091C</td>
<td>ETH_MACL3A3R</td>
<td>L3A30[31:0]</td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
</tbody>
</table>
Table 682. Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0930</td>
<td>ETH_MACL3L4C1R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0934</td>
<td>ETH_MACL4A1R</td>
<td>L4DP1[15:0]</td>
<td>L4SP1[15:0]</td>
</tr>
<tr>
<td>0x0938</td>
<td>ETH_MACA1IR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0940</td>
<td>ETH_MACA01IR</td>
<td>L3A01[31:0]</td>
<td></td>
</tr>
<tr>
<td>0x0944</td>
<td>ETH_MACA11IR</td>
<td>L3A11[31:0]</td>
<td></td>
</tr>
<tr>
<td>0x0948</td>
<td>ETH_MACA21R</td>
<td>L3A21[31:0]</td>
<td></td>
</tr>
<tr>
<td>0x094C</td>
<td>ETH_MACA31R</td>
<td>L3A31[31:0]</td>
<td></td>
</tr>
<tr>
<td>0x0950</td>
<td>ETH_MACSTSCR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B00</td>
<td>ETH_MACSSIR</td>
<td>SSINC[7:0]</td>
<td></td>
</tr>
<tr>
<td>0x0B04</td>
<td>ETH_MACSTSR</td>
<td></td>
<td>TSS[31:0]</td>
</tr>
<tr>
<td>0x0B08</td>
<td>ETH_MACSTNR</td>
<td></td>
<td>TSSS[30:0]</td>
</tr>
<tr>
<td>0x0B0C</td>
<td>ETH_MACSTNSUR</td>
<td></td>
<td>TSS[31:0]</td>
</tr>
<tr>
<td>0x0B10</td>
<td>ETH_MACSTNSUR</td>
<td></td>
<td>TSSS[30:0]</td>
</tr>
<tr>
<td>0x0B14</td>
<td>ETH_MACSTNSUR</td>
<td></td>
<td>TSS[31:0]</td>
</tr>
</tbody>
</table>

Note: The table shows the register map and reset values for the Ethernet MAC module. The reset values are provided for each register, and the table continues with additional registers and descriptions.
Table 682. Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset value</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0B18</td>
<td>ETH_MACTSAR</td>
<td></td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0B1C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B20</td>
<td>ETH_MACTSSR</td>
<td></td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0B2C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B30</td>
<td>ETH_MACTXTSSNR</td>
<td></td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0B3C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B40</td>
<td>ETH_MACACR</td>
<td></td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0B44</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B48</td>
<td>ETH_MACATSNR</td>
<td></td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0B4C</td>
<td>ETH_MACATSSR</td>
<td></td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0B50</td>
<td>ETH_MACTSIACR</td>
<td></td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0B54</td>
<td>ETH_MACTSEACR</td>
<td></td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0B58</td>
<td>ETH_MACTSICNR</td>
<td></td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0B5C</td>
<td>ETH_MACTSECNR</td>
<td></td>
<td>0x00000000000000000000000000000000</td>
</tr>
</tbody>
</table>
### Table 682. Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0B60 -</td>
<td>ETH_MACPPSCR</td>
<td>Reserved</td>
<td>0x0B70</td>
<td>ETH_MACPPSCR</td>
<td>(alternate)</td>
</tr>
<tr>
<td>0x0B6C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B70</td>
<td>ETH_MACPPSCR</td>
<td></td>
<td></td>
<td>ETH_MACPPSCR</td>
<td></td>
</tr>
<tr>
<td>0x0B70</td>
<td>(alternate)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B74 -</td>
<td>ETH_MACPSTTSR</td>
<td>Reserved</td>
<td>0x0B7C</td>
<td>ETH_MACPSIR</td>
<td>PPSINT[31:0]</td>
</tr>
<tr>
<td>0x0B7C</td>
<td>ETH_MACPSTNR</td>
<td></td>
<td></td>
<td>ETH_MACPSWR</td>
<td>PPSWIDTH[31:0]</td>
</tr>
<tr>
<td>0x0B80</td>
<td>ETH_MACPSCSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B84</td>
<td>ETH_MACPSTTSR</td>
<td>TSTRH0[31:0]</td>
<td></td>
<td>ETH_MACPSTNR</td>
<td>TTSL0[30:0]</td>
</tr>
<tr>
<td>0x0B84</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B88</td>
<td>ETH_MACPSCSR</td>
<td>Reserved</td>
<td>0x0B8C</td>
<td>ETH_MACPSCSR</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0B8C</td>
<td>ETH_MACPSTTSR</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B90 -</td>
<td>ETH_MACPSCSR</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B9C</td>
<td>ETH_MACPSTTSR</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0BC0</td>
<td>ETH_MACPSCSR</td>
<td>DDRDIS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0BC0</td>
<td></td>
<td>APDREQTRIG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0BC0</td>
<td></td>
<td>ASYNCTRIG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0BC0</td>
<td></td>
<td>ADPDREQEN</td>
<td></td>
<td></td>
<td>PTDGEN</td>
</tr>
<tr>
<td>0x0BC4</td>
<td>ETH_MACPSTTSR</td>
<td>SPI0[31:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0BC4</td>
<td>ETH_MACPSTTSR</td>
<td>SPI0[31:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Refer to *Section 2.3 on page 111* for the register boundary addresses.
58  Debug support (DBG)

58.1  Introduction

A comprehensive set of debug features is provided to support software development and system integration:

- Breakpoint debugging of the CPU core
- Code execution tracing
- Software instrumentation
- Cross-triggering

The debug features can be controlled via a JTAG/Serial-wire debug access port, using industry standard debugging tools. A trace port allows data to be captured for logging and analysis.

The debug features are based on Arm® CoreSight™ components.

- SWJ-DP: JTAG/Serial-wire debug port
- AHB-AP: AHB access port
- ROM table
- System control space (SCS)
- Breakpoint unit (BPU)
- Data watchpoint and trace unit (DWT)
- Instrumentation trace macrocell (ITM)
- Embedded Trace Macrocell™ (ETM)
- Cross trigger interface (CTI)
- Trace port interface unit (TPU)

The debug features are accessible by the debugger via the AHB-AP.

Additional information can be found in the Arm® documents referenced in Section 58.13.
58.2  DBG functional description

58.2.1  DBG block diagram

Figure 829. Block diagram of debug support infrastructure

58.2.2  DBG pins and internal signals

Table 684. JTAG/Serial-wire debug port pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>JTAG debug port</th>
<th>SW debug port</th>
<th>Pin assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Type</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>JTMS/SWDIO</td>
<td>I</td>
<td>JTAG test mode select</td>
<td>IO</td>
</tr>
<tr>
<td>JTCK/SWCLK</td>
<td>I</td>
<td>JTAG test clock</td>
<td>I</td>
</tr>
<tr>
<td>JTDI(1)</td>
<td>I</td>
<td>JTAG test data input</td>
<td>-</td>
</tr>
<tr>
<td>JTDI</td>
<td>O</td>
<td>JTAG test data output</td>
<td>-</td>
</tr>
<tr>
<td>nJTRST</td>
<td>I</td>
<td>JTAG test reset</td>
<td>-</td>
</tr>
</tbody>
</table>

1. TDI is hosted on the same IO as a USBPD-CC line. To avoid pull-up/down conflict, a user option can help to decide whether the pad is used as TDI or as CC.

Table 685. Trace port pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Type</th>
<th>Description</th>
<th>Pin assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACED0</td>
<td>O</td>
<td>Trace synchronous data out 0</td>
<td>Refer to the datasheet</td>
</tr>
<tr>
<td>TRACED1</td>
<td>O</td>
<td>Trace synchronous data out 1</td>
<td>Refer to the datasheet</td>
</tr>
<tr>
<td>TRACED2</td>
<td>O</td>
<td>Trace synchronous data out 2</td>
<td>Refer to the datasheet</td>
</tr>
<tr>
<td>TRACED3</td>
<td>O</td>
<td>Trace synchronous data out 3</td>
<td>Refer to the datasheet</td>
</tr>
<tr>
<td>TRACECK</td>
<td></td>
<td>Trace clock</td>
<td>Refer to the datasheet</td>
</tr>
</tbody>
</table>
58.2.3  **DBG reset and clocks**

The debug port (SWJ-DP) is reset by a power-on reset and when waking up from Standby mode.

The debugger supplies the clock for the debug port via the debug interface pin JTCK/SWCLK. This clock is used to register the serial input data in both serial-wire and JTAG modes, as well as to operate the state machines and internal logic of the debug port. This clock must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the DCLK domain that covers the rest of the SWJ-DP and the access port.

The DCLK is a gated version of the system clock.

The DCLK domain is enabled by the debugger using the CDBGPWRUPREQ bit in the **DP control and status register (DP_CTRL/STATR)**. The clock must be enabled before the debugger can access any of the debug features on the device. The availability of the clock is reflected in the CDBGPWRUPACK bit in DP_CTRL/STATR. The DCLK is disabled at power-up, and must be disabled when the debugger is disconnected, to avoid wasting energy.

The debug and trace components included in the processor are clocked with the processor clock.

58.2.4  **DBG power domains**

The debug components are located in the core power domain. This means that the debugger connection is not possible in Shutdown or Standby low-power mode. To avoid losing the connection when the device enters Standby mode, the power can be maintained to the core by setting a bit in the **DBGMCU configuration register (DBGMCU_CR)**. This also keeps the processor clocks active and holds off the reset, so that the debug session is maintained.

58.2.5  **Debug and low-power modes**

The devices include power saving features that allow the core power domain to be switched off or stopped when not required. If the power is switched off or if the core is not clocked, all debug components are inaccessible to the debugger. To avoid this, power-saving mode emulation is implemented. If the emulation is enabled for a domain, the domain still enters power-saving mode, but its clock and power are maintained. In other words, the domain behaves as if it is in power-saving mode, but the debugger does not lose the connection.

The emulation mode is programmed in the microcontroller debug (DBGMCU) unit. For more information refer to **Section 58.12: Microcontroller debug unit (DBGMCU)**.

### Table 686. Single-wire trace port pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Type</th>
<th>Description</th>
<th>Pin assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACESWO</td>
<td>O</td>
<td>Single-wire trace asynchronous data out</td>
<td>PB3(^{(1)})</td>
</tr>
</tbody>
</table>

1. TRACESWO is multiplexed with JTDO. This means that single-wire trace is only available when using the serial-wire debug interface, and not when using JTAG.
58.2.6 Security

The trace and debug components allow a high degree of access to the processor and system during product development. In order to protect user code and ensure that the debug features cannot be used to alter or compromise the normal operation of the finished product, these features can be disabled or limited in scope. For example, secure software debug and trace can be disabled without preventing the debug of non-secure code.

Debugger access to secure memory (when permitted) must be performed using secure transactions on the debug AHB, that is, with the PROT[6] bit set in the AP1 control/status word register (AP1_CSWR).

Debugger access is disabled while the processor is booting from system flash memory.

The following authentication signals are used by the system to determine which debug features are enabled or disabled:

- **dbgen**: global enable for all debug features
  - 0: All debug features are disabled.
  - 1: Debug features in non-secure state are enabled. Debug features in secure state are dependent on the state of the **spiden** signal.

- **spiden**: enables debug in secure state when **dbgen** = 1.
  - 0: Debug features are disabled in secure state.
  - 1: Debug features are enabled in secure state.

- **niden**: enables trace and performance monitoring (non-invasive debug).
  - 0: Trace generation is disabled.
  - 1: Trace generation in non-secure state is enabled. Trace generation in secure state is dependent on the state of the **spniden** signal.

- **spniden**: enables trace and performance monitoring in secure state when **niden** = 1.
  - 0: Trace generation is disabled in secure state.
  - 1: Trace generation is enabled in secure state.

For detailed information on the behavior of each component according to the state of the authentication signals, refer to the relevant chapter, or to the relevant Arm® technical documentation.

The state of the signals are set according to the debug state as shown in Table 687, when TrustZone is enabled (TZEN = 0xB4), and Table 688 when TrustZone is disabled (TZEN = 0xC3).

<table>
<thead>
<tr>
<th>Debug state</th>
<th>Authentication signal state</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>dbgen = 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>spiden = 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>niden = 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>spniden = 1</td>
<td>Debug and trace is enabled whatever the state of the processor. Debugger access to secure memory is permitted.</td>
</tr>
</tbody>
</table>

Table 687. Authentication signal states with TrustZone enabled (TZEN = 0xB4)
The state of the authentication signals can be read from the DAUTHSTATUS register in the system control space (SCS) of the Cortex®-M33.

The debug state depends on the product life cycle state (see chapter [product life cycle]), and the debug authentication state (see Section 58.2.7: Debug authentication).

### Table 687. Authentication signal states with TrustZone enabled (TZEN = 0xB4)

<table>
<thead>
<tr>
<th>Debug state</th>
<th>Authentication signal state</th>
<th>Description</th>
</tr>
</thead>
</table>
| CLOSED SECURE     | dbgen = 1  
                   spiden = 0  
                   niden = 1  
                   spniden = 0 | Debug and trace is enabled when the processor is in non-secure state. Debugger access to non-secure memory is permitted. Debugger access to secure memory is disabled. |
| CLOSED            | dbgen = 0  
                   spiden = 0  
                   niden = 0  
                   spniden = 0 | Debug and trace is disabled.                                                |

### Table 688. Authentication signal states with TrustZone disabled (TZEN = 0xC3)

<table>
<thead>
<tr>
<th>Debug state</th>
<th>Authentication signal state</th>
<th>Description</th>
</tr>
</thead>
</table>
| OPEN        | dbgen = 1  
                   spiden = 1  
                   niden = 1  
                   spniden = 1 | Debug and trace is enabled whatever the state of the processor. All memory and resources are considered non-secure and accessible to the debugger. |
| CLOSED      | dbgen = 0  
                   spiden = 0  
                   niden = 0  
                   spniden = 0 | Debug and trace is disabled.                                                |

The state of the authentication signals can be read from the DAUTHSTATUS register in the system control space (SCS) of the Cortex®-M33.

The debug state depends on the product life cycle state (see chapter [product life cycle]), and the debug authentication state (see Section 58.2.7: Debug authentication).

### Table 689. Life cycle state and debug states

<table>
<thead>
<tr>
<th>Product life cycle state</th>
<th>Debug state</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>TZ_CLOSED</td>
<td>CLOSED-SECURE</td>
</tr>
<tr>
<td>CLOSED (debug not authenticated)</td>
<td>CLOSED</td>
</tr>
<tr>
<td>CLOSED (debug constrained)</td>
<td>OPEN/CLOSED-SECURE(1)</td>
</tr>
<tr>
<td>LOCKED</td>
<td>CLOSED</td>
</tr>
</tbody>
</table>

1. Depends on authorization level.
58.2.7 Debug authentication

Figure 830. Product life cycle states and debug authentication

Figure 830 shows the product life cycle and debug authentication states.

If the device is in CLOSED (product life cycle) state, the debug state is CLOSED. The debug authentication procedure allows a trusted debugger to reopen access without compromising sensitive information called the Root Of Trust (ROT).

Reopening the debug is possible only if sensitive asset security is ensured (and TrustZone is enabled (TZEN = 0xB4). This is called Constrained Debug, as constraints ensure the security of the ROT information.

Alternatively, a partial or full regression mechanism can be used when security of sensitive information cannot be guaranteed. This is called Regression, as regression ensures removal of the sensitive information before reopening the debug.

- Partial regression corresponds to releasing non-secure code and assets. The intermediate state which allows partial regression management is called NS-Regression.
- Full regression corresponds to releasing all code and assets. The intermediate state allowing full regression management is called Regression.

Debug authentication control principle

The debug authentication is one of the most critical security features of the system considering that with a debugger the user can access a large part of the system.

To control re-opening of the debug, the device imposes a debug authentication protocol.
The protocol implements a challenge response mechanism based on asymmetric cryptography to authenticate the host. It relies on a key pair, with a Public Key stored in the device, and a Private Key from the host library, used to sign a random value (the challenge) generated by the device.

The protocol implements a bidirectional communication between the host and the device through a mailbox interface located in the DBGMCU.

The host can write to the mailbox via the JTAG/SWD interface. It expects to get responses and messages from the device via the same mailbox.

The debug authentication protocol is launched on a power-on reset of the device, when an “open request” message is posted by the host.

The protocol is based on:
- Initial message: posted by the host combined with a reset to launch the debug authentication process on the device.
- Challenge message: the device generates a random value, to be signed by the host, when sending back the response.
- Response: the host sends a message to the device proving its authenticity. This is done using a tool to generate a token.

The implementation is ensured by code embedded in the system flash. This code is called automatically after reset if an initial message has been posted by the host in the mailbox.

After a first sequence of mutual authentication to align on protocol version, type of device, and similar parameters, the device generates a random value that to be signed by the host with a private key when building the response.

The STMicroelectronics implementation is based on the Arm® PSA-ADAC solution for debug authentication.

The debug authentication can be implemented using a proprietary or open source protocol. As this feature is critical for security, the STM32H5 devices come with debug authentication provisioned in system flash. STMicroelectronics provides the host tools integrated with some debug environments (IDEs).

### Debug authentication provisioning

Debug authentication is natively supported by STM32H5 platform. It means that the data used by ST debug authentication (ST-DA) must be provisioned at a defined location in secure key storage area (OBKeys defined in flash memory).

The debug authentication configuration must be done only when the PRODUCT_STATE is “Provisioning”, cannot be performed when PRODUCT_STATE is “Open”.

The following data must be provisioned (refer to Section 7.7.2: RSS user functions):
- Data must be provisioned at the very beginning of the OBK_HDPL1 mapping (0x0FFD 0100) (see Section 7.5.2: OBK access per HDPL level)
- RSS_Lib encryption option must be set for STM32H57x devices
- RSS_Lib encryption option must be reset for STM32H56x devices

The data to provision are defined in Table 690, they depend upon the TZEN option byte setting (enabled or disabled).
Caution: Configure debug authentication as explained above before changing to states different from Open or Provisioning.

Table 690. Definition of data to provision

<table>
<thead>
<tr>
<th>Data size</th>
<th>TZEN enabled (certificates)</th>
<th>TZEN disabled (password: only full regression is allowed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 bytes</td>
<td>SHA256 of the overall blob (i.e. the two next fields)</td>
<td>SHA256 of the password</td>
</tr>
<tr>
<td>16 bytes</td>
<td>128-bis permission mask (see Table 691)</td>
<td>128 bits are reserved, to be set to 0</td>
</tr>
</tbody>
</table>

Table 691. Permission mask (Endianness: Little Endian)

<table>
<thead>
<tr>
<th>Bit(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0</td>
<td>When set it allows Intrusive debug from HDPL1 – NS</td>
</tr>
<tr>
<td>Bit 1</td>
<td>When set it allows Intrusive debug from HDPL2 – NS</td>
</tr>
<tr>
<td>Bit 2</td>
<td>When set it allows Intrusive debug from HDPL3 – NS</td>
</tr>
<tr>
<td>Bit 3</td>
<td>Reserved</td>
</tr>
<tr>
<td>Bit 4</td>
<td>When set it allows Intrusive debug from HDPL1 – S</td>
</tr>
<tr>
<td>Bit 5</td>
<td>When set it allows Intrusive debug from HDPL2 – S</td>
</tr>
<tr>
<td>Bit 6</td>
<td>When set it allows Intrusive debug from HDPL3 – S</td>
</tr>
<tr>
<td>Bits 7:11</td>
<td>Reserved</td>
</tr>
<tr>
<td>Bit 12</td>
<td>Regression to TZ-Closed, when 1 regression to TZ-Closed is allowed</td>
</tr>
<tr>
<td>Bit 13</td>
<td>Reserved</td>
</tr>
<tr>
<td>Bit 14</td>
<td>Regression (Full regression), when 1 full regression is allowed</td>
</tr>
<tr>
<td>Bit 15:127</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

58.3 Serial-wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a CoreSight™ component that implements an external access port for connecting debugging equipment.

Two types of interface can be configured:
- a 5-pin standard JTAG interface (JTAG-DP)
- a 2-pin (clock + data) serial-wire debug port (SW-DP)

These two modes are mutually exclusive, since they share the same IO pins.

By default, the JTAG-DP is selected after a system or a power-on reset. The five IO pins are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on JTDI, JTMS/SWDIO, and nJTRST, as well as a pull-down resistor on JTCK/SWCLK.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

... (50 or more ones) ..., 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, ... (50 or more ones) ...
JTCK/SWCLK must be cycled for each data bit.

In SW-DP mode, the unused JTAG pins JTDI, JTDO and nJTRST can be used for other functions.

**Note:** All SWJ port IOs can be reconfigured to other functions by software, but debugging is no longer possible.

### 58.3.1 JTAG debug port

The JTAG-DP implements a TAP state machine (TAPSM), shown in the figure below, based on IEEE Std 1149.1-1990. The state machine controls two scan chains, one associated with an instruction register (IR) and the other one with a number of data registers (DR).

---

**Figure 831. JTAG TAP state machine**

---
The operation of the JTAG-DP is as follows:

1. When the TAPSM goes through the Capture-IR state, 0b0001 is transferred to the instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.

2. While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that on the first tick:
   - The LSB of the IR scan chain is output on JTDO.
   - Bit[n] of the IR scan chain is transferred to bit[n-1].
   - The value on JTDI is transferred to the MSB of the IR scan chain.

3. When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred to the instruction register.

4. When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers to one of the DR scan chains, connected between JTDI and JTDO.

5. The value held in the instruction register determines which data register, and associated DR scan chain, are selected.

6. This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shifts in the Shift-IR state.

7. When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred to the selected data register.

8. When the TAPSM is in the Run-Test/Idle state, no special actions occurs. The IDCODE instruction is loaded in IR.

When active, the nJTRST signal resets the state machine asynchronously to the test-logic-reset state.

The data registers corresponding to the 4-bit IR instructions are listed in the table below.

### Table 692. JTAG-DP data registers

<table>
<thead>
<tr>
<th>IR instruction</th>
<th>DR register</th>
<th>Scan chain length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 to 0111 (BYPASS)</td>
<td>1</td>
<td>Not implemented: BYPASS selected</td>
<td></td>
</tr>
</tbody>
</table>
| 1000 | ABORT | 35 | ABORT register
   - bits 34:3 = APABORT[31:0]: Write 0x0000 0001 to abort an ongoing access port transaction
   - bits 2:1 = A[3:2] = 00
   - bit 0 = RnW = 0: write only |
| 1001 (BYPASS) | 1 | Reserved: BYPASS selected |
The DR registers are described in more detail in the Arm® Debug Interface Architecture Specification (see Section 58.13).

### 58.3.2 Serial-wire debug port

The serial-wire debug protocol uses the following pins:
- **SWCLK**: clock from host to target
- **SWDIO**: bi-directional serial data (100 kΩ pull-up required)

Serial data is transferred LSB first, synchronously with the clock.

A transfer comprises three phases:

<table>
<thead>
<tr>
<th>IR instruction</th>
<th>DR register</th>
<th>Scan chain length</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1010           | DPACC       | 35                | Debug port access register  
Initiates the debug port and gives access to a debug port register.  
– When transferring data IN:  
  bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request  
  bits 2:1 = A[3:2] = 2-bit address of a debug port register  
  bit 0 = RnW = read request (1) or write request (0)  
– When transferring data OUT:  
  bits 34:3 = DATA[31:0] = 32-bit data read following a read request  
  bits 2:0 = ACK[2:0] = 3-bit Acknowledge:  
    – 010 = OK/FAULT  
    – 001 = WAIT  
    – others = reserved |
| 1011           | APACC       | 35                | Access port access register  
Initiates an access port and gives access to an access port register.  
– When transferring data IN:  
  bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request  
  bits 2:1 = A[3:2] = 2-bit sub-address of an access port register  
  bit 0 = RnW = Read request (1) or write request (0)  
– When transferring data OUT:  
  bits 34:3 = DATA[31:0] = 32-bit data read following a read request  
  bits 2:0 = ACK[2:0] = 3-bit Acknowledge:  
    – 010 = OK/FAULT  
    – 001 = WAIT  
    – others = reserved |
| 1100           | (BYPASS)    | 1                 | Reserved: BYPASS selected |
| 1101           | (BYPASS)    | 1                 | Reserved: BYPASS selected |
| 1110           | IDCODE      | 32                | Identification code  
0x6BA0 0477: Cortex®-M33 JTAG debug port ID code |
| 1111           | BYPASS      | 1                 | Bypass  
A single JTCK cycle delay is inserted between JTDI and JTDO. |
1. packet request (8 bits) transmitted by the host (see Table 693)
2. acknowledge response (3 bits) transmitted by the target (see Table 694)
3. data transfer (33 bits) transmitted by the host (in case of a write) or target (in case of a read) (see Table 695)

The data transfer only occurs if the acknowledge response is OK.

Between each phase, if the direction of the data is reversed, a single clock-cycle turn-around time is inserted.

### Table 693. Packet request

<table>
<thead>
<tr>
<th>Bit field</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Start</td>
<td>Must be 1</td>
</tr>
</tbody>
</table>
| 1         | APnDP | – 0: DP register access - see Section 58.3.3: Debug port registers  
            |       | – 1: AP register access - see Section 58.4: Access ports |
| 2         | RnW   | – 0: write request                   |
|           |       | – 1: read request                    |
| 4:3       | A(3:2)| Address field of the DP or AP register (refer to Table 697 or Table 699) |
| 5         | Parity| Single bit parity of preceding bits  |
| 6         | Stop  | 0                                    |
| 7         | Park  | Not driven by host, must be read as 1 by target |

### Table 694. ACK response

<table>
<thead>
<tr>
<th>Bit field</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2:0       | ACK  | – 000: FAULT  
            |       | – 010: WAIT  
            |       | – 100: OK    |

### Table 695. Data transfer

<table>
<thead>
<tr>
<th>Bit field</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>WDATA or RDATA</td>
<td>Write or read data</td>
</tr>
<tr>
<td>32</td>
<td>Parity</td>
<td>Single-bit parity of 32 data bits</td>
</tr>
</tbody>
</table>

In the case of a FAULT or WAIT ACK response from the target, the data transfer phase is canceled, unless overrun detection is enabled: in this case, the data is ignored by the target (in the case of a write), or not driven (in the case of a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists in 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the serial-wire debug protocol, refer to the Arm® Debug Interface Architecture Specification [1].

**Note:** The SWJ-DP implements SWD protocol version 2.
58.3.3 Debug port registers

Both the SW-DP and the JTAG-DP access the debug port (DP) registers listed in Table 697. The debugger can access the DP registers as follows:

1. Program the A(3:2) field in the DPACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the data field with the write data. If using SWD, the A(3:2) and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit reset (see Table 693). The write data are sent in the data phase.

2. To access one of the banked DP registers at address 0x4, the register number must first be written to the DP_SELECTR register at address 0x8. Any subsequent read or write to address 0x4 accesses the register corresponding to the contents of the DP_SELECTR register.

### Table 696. Debug port registers

<table>
<thead>
<tr>
<th>Address</th>
<th>A(3:2) value</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>00</td>
<td>R</td>
<td>DP debug port identification register (DP_DPIDR) contains the IDCODE for the debug port.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td>DP abort register (DP_ABORTR)(^{(1)}) aborts the current AP transaction. This register is also used to clear the error flags in the DP_CTRLSTATR register.</td>
</tr>
<tr>
<td>0x4</td>
<td>01</td>
<td>R/W</td>
<td>If DP_SELECTR.DPBANKSEL[3:0] = 0x0, DP control and status register (DP_CTRLSTATR) controls the DP and provides status information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If DP_SELECTR.DPBANKSEL[3:0] = 0x1, DP data link control register (DP_DLCR)(^{(2)}) controls the operating mode of the SWD data link.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If DP_SELECTR.DPBANKSEL[3:0] = 0x2, DP target identification register (DP_TARGETIDR) provides target identification information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If DP_SELECTR.DPBANKSEL[3:0] = 0x3, DP data link protocol identification register (DP_DLPIDR)(^{(2)}) provides the SWD protocol version.</td>
</tr>
<tr>
<td>0x8</td>
<td>10</td>
<td>R</td>
<td>DP event status register (DP_RESENDR)(^{(2)}) returns the value that was returned by the last AP read or DP_RDBUFF read. Used in the event of a corrupted read transfer.</td>
</tr>
<tr>
<td></td>
<td>W</td>
<td></td>
<td>DP access port select register (DP_SELECTR) selects the access port, access port register bank, and DP register at address 0x4.</td>
</tr>
<tr>
<td>0xC</td>
<td>11</td>
<td>R</td>
<td>DP read buffer register (DP_RDBUFFFF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Via JTAG-DP, this register allows the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Via SW-DP, this register contains the result of the preceding AP read access, allowing a new AP access to be avoided.</td>
</tr>
</tbody>
</table>

1. Access to the AP ABORT register from the JTAG-DP is done using the ABORT instruction.
2. Only accessible via SW-DP. Register is “reserved” via JTAG-DP.
DP debug port identification register (DP_DPIDR)

Address offset: 0x0
Reset value: 0x6BA0 2477

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:28 **REVISION[3:0]**: revision code
- 0x6: Rev 6

Bits 27:20 **PARTNO[7:0]**: part number for the debug port
- 0xBA

Bits 19:17 Reserved, must be kept at reset value.
- Bit 16 **MIN**: minimal debug port (MINDP) implementation
  - 0x0: MINDP not implemented

Bits 15:12 **VERSION[3:0]**: debug port architecture version
- 0x2: DPv2

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
- 0x23B: Arm® JEDEC code

Bit 0 Reserved, must be kept at reset value.

DP abort register (DP_ABORTR)

Address offset: 0x0
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

w w w w
Bits 31:5  Reserved, must be kept at reset value.

Bit 4  **ORUNERRCLR**: overrun error clear
0: no effect
1: STICKYORUN bit cleared in DP_CTRL/STATR register

Bit 3  **WDERRCLR**: write data error clear
0: no effect
1: WDATAERR bit cleared in DP_CTRL/STATR register

Bit 2  **STKERRCLR**: sticky error clear
0: no effect
1: STICKYERR bit cleared in DP_CTRL/STATR register

Bit 1  Reserved, must be kept at reset value.

Bit 0  **DAPABORT**: current AP transaction aborted if excessive number of WAIT responses returned
This bit indicates that the transaction is stalled.
0: no effect
1: transaction aborted

**DP control and status register (DP_CTRLSTATR)**

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:30  Reserved, must be kept at reset value.

Bit 29  **CDBGPWRUPACK**: debug power-up acknowledge
See description in *Section 58.2.5: Debug and low-power modes*.
0: DCLK gated
1: DCLK enabled

Bit 28  **CDBGPWRUPREQ**: debug power-up request
This bit controls the DCLK enable request signal.
0: requests DCLK gating
1: requests DCLK enable
Bits 27:8  Reserved, must be kept at reset value.

Bit 7  **WDATAERR**: write data error (read-only) in SW-DP
This bit indicates that there is a parity or framing error on the data phase of a write, or a write accepted by the DP is then discarded without being submitted to the AP. This bit is reset by writing 1 to the ABORT.WDERRCLR bit.

0: no error
1: an error occurred

*Note: This bit is reserved in JTAG-DP.*

Bit 6  **READOK**: AP read response (read-only) in SW-DP
This bit indicates the response to the last AP read access.

0: read not OK
1: read OK

*Note: This bit is reserved in JTAG-DP.*

Bit 5  **STICKYERR**: transaction error (read-only in SW-DP, read/write in JTAG-DP)
This bit indicates that an error occurred in an AP transaction. It is reset by writing 1 to the DP_ABORTR.STKERRCLR bit (in SW-DP and JTAG-DP)

0: no error
1: an error occurred

Bits 4:2  Reserved, must be kept at reset value.

Bit 1  **STICKYORUN**: overrun (read-only in SW-DP, read/write in JTAG-DP).
This bit indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set. It is reset by writing 1 to the DP_ABORTR.ORUNERRCLR bit (in SW-DP and JTAG-DP).

0: no overrun
1: an overrun occurred

Bit 0  **ORUNDETECT**: overrun detection mode enable.

0: disabled
1: enabled. In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until the STICKYORUN bit is cleared.

---

**DP data link control register (DP_DLCR)**

Address offset: 0x4
Reset value: 0x0000 0000 0000

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x1.
Debug support (DBG)

DP target identification register (DP_TARGETIDR)

Address offset: 0x4
Reset value: 0xFFFF 0041

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x2.

| Bits 31:10 | Reserved, must be kept at reset value. |
| Bits 9:8   | TURNROUND[1:0]: tristate period for SWDIO |
|           | 0x0: 1 data bit period |
| Bits 7:6   | WIREMODE[1:0]: SW-DP mode |
|           | 0x0: synchronous mode |
| Bits 5:0   | Reserved, must be kept at reset value. |

DP data link protocol identification register (DP_DLPIDR)

Address offset: 0x4
Reset value: 0xFFFF 0001

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x3.

| Bits 31:28 | TREVISION[3:0]: target revision |
| Bits 27:12 | TPARTNO[15:0]: target part number |
|           | 0x4840: STM32H5 |
| Bits 11:1  | TDESIGNER[10:0]: target designer JEDEC code |
|           | 0x020: STMicroelectronics |
| Bit 0      | Reserved, must be kept at reset value. |
Bits 31:28  **TINSTANCE[3:0]**: target instance number
this field defines the instance number for the device in a multi-drop system.
0x0: instance number 0

Bits 27:4  Reserved, must be kept at reset value.

Bits 3:0 **PROT SVN[3:0]**: Serial-wire debug protocol version
0x1: version 2

### DP event status register (DP_RESE ND R)

Address offset: 0x8
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **RESEND[31:0]**: value returned by the last AP read or DP_RDBUFF read
This register is used in the event of a corrupted read transfer.

### DP access port select register (DP_SELECT R)

Address offset: 0x8
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:24  **APSEL[7:0]**: access port select
This field selects the access port for the next transaction.
0x00: AP0 - System debug access port (APB-AP)
0x01: AP1 - Cortex®-M33 debug access port (AHB-AP)
Others: reserved
58.3.4 Debug port register map and reset values

These registers are not on the CPU memory bus, they are accessed only through SW-DP and JTAG-DP debug interface.

The debug port address offset is 4-bit wide, where the two most significant bits are defined in the JTAG-DP register DPACC or SW-DP packet request A[3:2] field. The two least significant bits are 00.

Table 697. Debug port register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>24</th>
<th>32</th>
<th>30</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>DP_DPIDR</td>
<td>REV [3:0]</td>
<td>PARTNO[7:0]</td>
<td>MIN</td>
<td>VERSION [3:0]</td>
<td>DESIGNER[10:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

2960/3119 RM0481 Rev 1
There are two access ports (AP) attached to the DP.

- System debug access port (AP0): Enables access to the DBGMCU and the system ROM table via an APB bus.
- Cortex®-M33 debug access port (AP1): Enables access to the debug and trace features integrated in the Cortex®-M33 processor core via its internal AHB bus.
58.4.1 Access port registers

The access ports are of type MEM-AP: the debug and trace component registers are mapped in the address space of the AHB. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in Table 699.

The address of the AP registers is composed of the following fields:

- bits [7:4]: content of the APBANKSEL[3:0] field in the DP access port select register (DP_SELECTR)
- bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see Table 692), or of the SW-DP packet request (see Table 693), depending on the debug interface used
- bits [1:0]: always set to 0

The content of the DP_SELECTR.APSEL[3:0] field defines which MEM-AP is being accessed.

<table>
<thead>
<tr>
<th>Address</th>
<th>APBANKSEL</th>
<th>A(3:2)</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0x0</td>
<td>0</td>
<td>CSWR</td>
<td>Control/status word register</td>
</tr>
<tr>
<td>0x04</td>
<td>0x0</td>
<td>1</td>
<td>TAR</td>
<td>Transfer address register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Target address for the bus transaction.</td>
</tr>
<tr>
<td>0x08</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>- Reserved</td>
</tr>
<tr>
<td>0x0C</td>
<td>0x0</td>
<td>3</td>
<td>DRWR</td>
<td>Data read/write register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Access to this register triggers a corresponding transaction on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the debug bus to the address in TAR[31:0]</td>
</tr>
<tr>
<td>0x10</td>
<td>0x1</td>
<td>0</td>
<td>BD0R</td>
<td>Banked data 0 register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Access to this register triggers a corresponding transaction on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the debug bus to the address in TAR[31:4] + 0x0.</td>
</tr>
<tr>
<td>0x14</td>
<td>0x1</td>
<td>1</td>
<td>BD1R</td>
<td>Banked data 1 register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Access to this register triggers a corresponding transaction on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the debug bus to the address in TAR[31:4] + 0x4.</td>
</tr>
<tr>
<td>0x18</td>
<td>0x1</td>
<td>2</td>
<td>BD2R</td>
<td>Banked data 2 register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Access to this register triggers a corresponding transaction on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the debug bus to the address in TAR[31:4] + 0x8.</td>
</tr>
<tr>
<td>0x1C</td>
<td>0x1</td>
<td>3</td>
<td>BD3R</td>
<td>Banked data 3 register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Access to this register triggers a corresponding transaction on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the debug bus to the address in TAR[31:4] + 0xC.</td>
</tr>
<tr>
<td>0x20</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>- Reserved</td>
</tr>
<tr>
<td>0x24 to 0xEC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>- Reserved</td>
</tr>
</tbody>
</table>
The debugger can access the AP registers as follows:

1. Program the APSEL[3:0] field in the DP access port select register (DP_SELECTR) to choose the AP, and the APBANKSEL[3:0] field in DP_SELECTR to select the register bank to be accessed.

2. Program the A(3:2) field in the APACC data register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit set (see Table 693). The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the AP registers (using the above AP register access procedure) as follows:

1. Program the transaction target address in the APx transfer address register (APx_TAR) (x = 0, 1).

2. Program the AP1 control/status word register (AP1_CSWR), if necessary, with the transfer parameters (AddrInc for example).

3. Write to or read from the APx data read/write register (APx_DRWR) (x = 0, 1) to initiate a bus transaction at the address held in AP_TAR. Alternatively, a read or write to the APx banked data n register (APx_BDnR) (x = 0, 1) triggers an access to TAR[31:4] + n address, allowing up to four consecutive addresses to be accessed without changing the address in the AP_TAR register.

For more detailed information on the MEM-AP refer to the Arm® Debug Interface Architecture Specification [1].

### AP0 control/status word register (AP0_CSWR)

Address offset: 0x0

Reset value: 0x8000 0042

<table>
<thead>
<tr>
<th>Address</th>
<th>APBANKSEL</th>
<th>A(3:2)</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF0</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xF4</td>
<td>0xF</td>
<td>1</td>
<td>CFGR</td>
<td>Configuration register (read only)</td>
</tr>
<tr>
<td>0xF8</td>
<td>0xF</td>
<td>2</td>
<td>BASE</td>
<td>Debug base address register (read only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td>Base address of the ROM table</td>
</tr>
<tr>
<td>0xFC</td>
<td>0xF</td>
<td>3</td>
<td>IDR</td>
<td>Identification register (read only)</td>
</tr>
</tbody>
</table>

Table 698. MEM-AP registers (continued)
AP1 control/status word register (AP1_CSWR)

Address offset: 0x0
Reset value: 0x4300 002

Bit 31 DBGSWEN: software access enable
Enables or disables software access to the APB bus
0: disable software access
1: enable software access

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:8 MODE[3:0]: mode of operation
0b0000: normal download or upload
other: reserved

Bit 7 TRINPROG: transfer in progress (read only)
This field indicates whether a transfer is currently in progress on the APB master port
0: no APB transfer in progress
1: APB transfer in progress

Bit 6 DEVICEEN: device enable status (read only)
1: APB transfers always enabled

Bits 5:4 ADDRINC[1:0]: auto-increment mode
Defines whether TAR address is automatically incremented after a transaction.
0x0: no auto-increment
0x1: address incremented by the size in bytes of the transaction (SIZE field)
other: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 SIZE[2:0]: size of next memory access transaction
0x2: word (32-bit)
Bit 31  Reserved, must be kept at reset value.

Bit 30  **SPROT**: secure transfer request
This field sets the protection attribute HPROT[6] of the bus transfer.
0: secure transfer, HPROT[6] = low
1: non-secure transfer, HPROT[6] = high
If SPIDEN = 0 and SPROT = 0, no bus transfer occurs

Bits 29:28  Reserved, must be kept at reset value.

Bits 27:24  **PROT[3:0]**: bus transfer protection
This field sets the protection attributes HPROT[3:0] of the bus transfer.
0bXXX0: instruction access
0bXXX1: data access
0bXXOX: user mode
0bXX1X: privilege mode
0bX0XX: non-bufferable
0bX1XX: bufferable
0b0XXX: non-shareable, no look-up, non-modifiable
0b1XXX: shareable, look-up, modifiable

Bit 23  **SPISTATUS**: secure debug authentication status
This field indicates the state of the SPIDEN signal
0: No secure AHB transfers allowed
1: Secure AHB transfers allowed

Bits 22:8  Reserved, must be kept at reset value.

Bit 7  **TRINPROG**: transfer in progress (read only)
This field indicates whether a transfer is currently in progress on the APB master port
0: No AHB transfer in progress
1: AHB transfer in progress

Bit 6  **DBGSTATUS**: debug enable (DBGEN) status
0: AHB transfers blocked
1: AHB transfers enabled

Bits 5:4  **ADDRINC[1:0]**: auto-increment mode
Defines whether TAR address is automatically incremented after a transaction.
0x0: no auto-increment
0x1: address incremented by the size of the transaction (SIZE field), in bytes. Single transfer.
0x2: address incremented by the size of the transaction (SIZE field), in bytes. Packs four 8-bit transfers or two 16-bit transfers into a 32-bit DAP transfer. Multiple transactions are carried out on the AHB interface.
Others: reserved

Bit 3  Reserved, must be kept at reset value.

Bits 2:0  **SIZE[2:0]**: size of next memory access transaction
0x0: byte (8-bit)
0x1: halfword (16-bit)
0x2: word (32-bit)
Others: reserved
APx transfer address register (APx_TAR) (x = 0, 1)
Address offset: 0x04
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 TA[31:0]: address of current transfer. In AP0, TA[1:0] are fixed at 0.

APx data read/write register (APx_DRWR) (x = 0, 1)
Address offset: 0x0C
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 TD[31:0]: data of current transfer

APx banked data n register (APx_BDnR) (x = 0, 1)
Address offset: 0x10 + 4 * n, (n = 0 to 3)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 DATA[31:0]: banked data of current transfer to address TA[31:4] + 4 * n. Auto address incrementing is not performed on APx_BD0-3R. Banked transfers are only supported for word transfers.
### APx base address register (APx_BASER) (x = 0, 1)

Address offset: 0xF8

Reset value: AP 0: 0xE00E 0003

Reset value: AP 1: 0xE00F E003

<table>
<thead>
<tr>
<th>BASEADDR[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BASEADDR[15:12]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r</td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
</tr>
</tbody>
</table>

Bits 31:12 **BASEADDR[31:12]**: base address (bits 31 to 12) of the first ROM table

The 12 LSBs are zero since the ROM table must be aligned on a 4-Kbyte boundary.

- 0xE00E0: AP0
- 0xE00FE: AP1

Bits 11:2 Reserved, must be kept at reset value.

- Bit 1 **FORMAT**: base-address register format
  - 1: Arm® debug interface v5

- Bit 0 **ENTRYPRESENT**: debug components presence
  - Indicates that debug components are present on the access port bus.
  - 1: debug components present

### APx identification register (APx_IDR) (x = 0, 1)

Address offset: 0xFC

Reset value: AP 0: 0x5477 0002

Reset value: AP 1: 0x8477 0001

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r r r r r</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**ST**

RM0481 Rev 1 2967/3119
58.4.2 Access port register map

These registers are not on the CPU memory bus, they are only accessed through SW-DP and JTAG-DP debug interfaces.


Table 699. Access port register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>AP0_CSWR</td>
<td>0x00</td>
<td>AP1_CSWR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x08</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x1C</td>
</tr>
</tbody>
</table>

Bits 31:28 **REVISION[3:0]**: revision number
- 0x5: r1p0
- 0x8: r0p9

Bits 27:24 **JEDECBANK[3:0]**: JEDEC bank
- 0x4: Arm®

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code
- 0x3B: Arm®

Bits 16:13 **CLASS[3:0]**:
- 0x1: MEM-AP

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**:
- 0x1: AHB-AP
- 0x2: APB-AP

<table>
<thead>
<tr>
<th>Bits 31:28</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>REVISION[3:0]</td>
<td>0x5:</td>
</tr>
<tr>
<td>0x01</td>
<td>REVISION[3:0]</td>
<td>0x8:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 27:24</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>JEDECBANK[3:0]</td>
<td>0x4:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 23:17</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>JEDECCODE[6:0]</td>
<td>0x3B:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 16:13</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>CLASS[3:0]</td>
<td>0x1:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 12:8</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 7:0</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>IDENTITY[7:0]</td>
<td>0x1:</td>
</tr>
</tbody>
</table>

Table 699. Access port register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>AP0_CSWR</td>
<td>0x00</td>
<td>AP1_CSWR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x08</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x1C</td>
</tr>
</tbody>
</table>

Reset value for each register is shown in the table.
The ROM table is a CoreSight™ component that contains the base addresses of the CoreSight™ debug components accessible via the access port to which it is attached. These tables allow a debugger to discover the topology of the CoreSight™ system automatically.

There is one top level ROM table behind each access port, APn. The base address of this ROM table can be obtained by reading the APn_BASER register of the access port. The top level ROM table may point in turn to other ROM tables.

The system ROM table is pointed to by the AP0 base register, AP0_BASER. It contains the base address pointer for the DBGMCU.

The system ROM table occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00E 0000 to 0xE00E 0FFC, when accessed by the debugger. It can be accessed by the CPU at address range 0x4402 0000 to 0x4402 0FFC.
There are two ROM tables in the CPU sub-system. The MCU ROM table is pointed to by the AP1 base register, AP1_BASER. It contains the base-address pointer for the processor ROM table and for the TPIU registers.

The MCU ROM table (see the table below) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F E000 to 0xE00F EFFC.

<table>
<thead>
<tr>
<th>Address offset in ROM table</th>
<th>Component name</th>
<th>Component base address</th>
<th>Component address offset</th>
<th>Size (Kbytes)</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>DBGMCU</td>
<td>0xE00E 4000 (debugger)</td>
<td>0x0000 4000</td>
<td>4</td>
<td>0x0000 4003</td>
</tr>
<tr>
<td>0x004</td>
<td>Top of table</td>
<td></td>
<td></td>
<td></td>
<td>0x0000 0000</td>
</tr>
<tr>
<td>0x008 to 0xFC8</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td>0x0000 0000</td>
</tr>
<tr>
<td>0xFC to 0xFFC</td>
<td>ROM table registers</td>
<td></td>
<td></td>
<td></td>
<td>See Table 703</td>
</tr>
</tbody>
</table>

There are two ROM tables in the CPU sub-system. The MCU ROM table is pointed to by the AP1 base register, AP1_BASER. It contains the base-address pointer for the processor ROM table and for the TPIU registers.

The MCU ROM table (see the table below) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F E000 to 0xE00F EFFC.

The processor ROM table contains the base-address pointer for the system control space (SCS) registers, that allow the debugger to identify the CPU core, as well as for the BPU, DWT, ITM, ETM and CTI.

The processor ROM table (see the table below) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F F000 to 0xE00F FFFC.

<table>
<thead>
<tr>
<th>Address in ROM table</th>
<th>Component name</th>
<th>Component base address</th>
<th>Component address offset</th>
<th>Size (Kbytes)</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE00F F000</td>
<td>SCS</td>
<td>0xE00E E000</td>
<td>0xFFF0 F000</td>
<td>4</td>
<td>0xFFF0 F003</td>
</tr>
<tr>
<td>0xE00F F004</td>
<td>DWT</td>
<td>0xE000 1000</td>
<td>0xFFF0 2000</td>
<td>4</td>
<td>0xFFF0 2003</td>
</tr>
<tr>
<td>0xE00F F008</td>
<td>BPU</td>
<td>0xE000 2000</td>
<td>0xFFF0 3000</td>
<td>4</td>
<td>0xFFF0 3003</td>
</tr>
<tr>
<td>0xE00F F00C</td>
<td>ITM</td>
<td>0xE000 0000</td>
<td>0xFFF0 1000</td>
<td>4</td>
<td>0xFFF0 1003</td>
</tr>
</tbody>
</table>

Table 700. System ROM table

Table 701. MCU ROM table

Table 702. Processor ROM table
# Table 702. Processor ROM table (continued)

<table>
<thead>
<tr>
<th>Address in ROM table</th>
<th>Component name</th>
<th>Component base address</th>
<th>Component address offset</th>
<th>Size (Kbytes)</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE00F F010</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0xFFF4 1002</td>
</tr>
<tr>
<td>0xE00F F014</td>
<td>ETM</td>
<td>0xE004 1000</td>
<td>0xFFF4 2000</td>
<td>4</td>
<td>0xFFF4 2003</td>
</tr>
<tr>
<td>0xE00F F018</td>
<td>CTI</td>
<td>0xE004 2000</td>
<td>0xFFF4 3000</td>
<td>4</td>
<td>0xFFF4 3003</td>
</tr>
<tr>
<td>0xE00F F01C</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0xFFF4 4002</td>
</tr>
<tr>
<td>0xE00F F020</td>
<td>Top of table</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x0000 0000</td>
</tr>
<tr>
<td>0xE00F F024 to 0xE00F FFC8</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x0000 0000</td>
</tr>
<tr>
<td>0xE00F FFCC to 0xE00F FFFC</td>
<td>ROM table registers</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>See Table 705</td>
</tr>
</tbody>
</table>
The topology for the CoreSight™ components is shown in the figure below.

**Figure 832. CoreSight topology**
58.5.1 System ROM table registers

System ROM memory type register (SYSROM_MEMTYPER)
Address offset: 0xFCC
Reset value: 0x0000 0001

| Bits 31:1 | Reserved, must be kept at reset value. |
| Bit 0   | SYSMEM: system memory |
| 0x1: system memory present on this bus |

System ROM CoreSight peripheral identity register 4 (SYSROM_PIDR4)
Address offset: 0xFD0
Reset value: 0x0000 0000

| Bits 31:8 | Reserved, must be kept at reset value. |
| Bits 7:4 | SIZE[3:0]: register file size |
| 0x0: The register file occupies a single 4-Kbyte region. |
| Bits 3:0 | JEP106CON[3:0]: JEP106 continuation code |
| 0x0: STMicroelectronics JEDEC continuation code |
### System ROM CoreSight peripheral identity register 0 (SYSROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00XX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]

- 0x84: STM32H563/H573 and STM32H562

### System ROM CoreSight peripheral identity register 1 (SYSROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 000X

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]

- 0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]

- 0x4: STM32H563/H573 and STM32H562
System ROM CoreSight peripheral identity register 2 (SYSROM_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 000A

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

| REV| JSEP| JEP106| |
|----|-----|-------|
| r   | r   | r     |

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:4  **REV[3:0]** component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: designer identification specified by JEDEC

0x2: STMicroelectronics JEDEC code

System ROM CoreSight peripheral identity register 3 (SYSROM_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>REV</th>
<th>CMOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:4  **REV[3:0]** metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]** customer modified
0x0: No customer modifications
**System ROM CoreSight component identity register 0 (SYSROM_CIDR0)**

Address offset: 0xFF0

Reset value: 0x0000 000D

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[7:0]**: component identification bits [7:0]

- 0x0D: Common identification value

**System ROM CoreSight peripheral identity register 1 (SYSROM_CIDR1)**

Address offset: 0xFF4

Reset value: 0x0000 0010

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **CLASS[3:0]**: Component identification bits [15:12] - component class

- 0x1: ROM table component


- 0x0: Common identification value
System ROM CoreSight component identity register 2 (SYSROM_CIDR2)
Address offset: 0xFF8
Reset value: 0x0000 0005

Bits 31:8 Reserved, must be kept at reset value.
Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
  0x05: common identification value

System ROM CoreSight component identity register 3 (SYSROM_CIDR3)
Address offset: 0xFFC
Reset value: 0x0000 00B1

Bits 31:8 Reserved, must be kept at reset value.
Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
  0xB1: Common identification value

58.5.2 System ROM table register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PREAMBLE[19:12]</td>
<td>r r r r r r r r r r r r</td>
</tr>
<tr>
<td></td>
<td>PREAMBLE[27:20]</td>
<td>r r r r r r r r r r r r</td>
</tr>
</tbody>
</table>

Table 703. System ROM table register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFCC</td>
<td>SYSROM_MEMTYPE</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0xFD0</td>
<td>SYSROM_PIDR4</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>
Refer to Table 700: System ROM table for register boundary addresses.

58.5.3 MCU ROM table registers

**MCU ROM memory type register (MCUROM_MEMTYPEPER)**

Address offset: 0xFCC

Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>bits 31:1</th>
<th>bits 16:15</th>
<th>bits 12:11</th>
<th>bits 8:7</th>
<th>bits 4:3</th>
<th>bits 0:1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFE0</td>
<td>SYSROM_PIDR0</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFE4</td>
<td>SYSROM_PIDR1</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFE8</td>
<td>SYSROM_PIDR2</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFEC</td>
<td>SYSROM_PIDR3</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFF0</td>
<td>SYSROM_CIDR0</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFF4</td>
<td>SYSROM_CIDR1</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFF8</td>
<td>SYSROM_CIDR2</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFFC</td>
<td>SYSROM_CIDR3</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: system memory

0x1: system memory present on this bus
**MCU ROM CoreSight peripheral identity register 4 (MCUROM_PIDR4)**

Address offset: 0xFD0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **SIZE[3:0]**: register file size  
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0  **JEP106CON[3:0]**: JEP106 continuation code  
0x0: STMicroelectronics JEDEC continuation code

**MCU ROM CoreSight peripheral identity register 0 (MCUROM_PIDR0)**

Address offset: 0xFE0  
Reset value: 0x0000 00XX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PARTNUM[7:0]**: part number bits [7:0]  
0x84: STM32H563/H573 and STM32H562
### MCU ROM CoreSight peripheral identity register 1 (MCUROM_PIDR1)

**Address offset:** 0xFE4  
**Reset value:** 0x0000 000X

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

- **Bits 31:8:** Reserved, must be kept at reset value.
- **Bits 7:4:** JEP106ID[3:0]: JEP106 identity code bits [3:0]  
  - 0x0: STMicroelectronics JEDEC code
- **Bits 3:0:** PARTNUM[11:8]: part number bits [11:8]  
  - 0x4: STM32H563/H573 and STM32H562

### MCU ROM CoreSight peripheral identity register 2 (MCUROM_PIDR2)

**Address offset:** 0xFE8  
**Reset value:** 0x0000 000A

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

- **Bits 31:8:** Reserved, must be kept at reset value.
- **Bits 7:4:** REVISION[3:0]: component revision number  
  - 0x0: rev r0p0
- **Bit 3:** JEDEC: JEDEC assigned value  
  - 0: designer identification specified by JEDEC
- **Bits 2:0:** JEP106ID[6:4]: JEP106 identity code bits [6:4]  
  - 0x2: STMicroelectronics JEDEC code

### MCU ROM CoreSight peripheral identity register 3 (MCUROM_PIDR3)

**Address offset:** 0xFEC  
**Reset value:** 0x0000 0000
MCU ROM CoreSight component identity register 0 (MCUROM_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  REVAND[3:0]: metal fix version
0x0: No metal fix

Bits 3:0  CMOD[3:0]: customer modified
0x0: No customer modifications

MCU ROM CoreSight peripheral identity register 1 (MCUROM_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 0010

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  PREAMBLE[7:0]: component identification bits [7:0]
0x0D: Common identification value

MCU ROM CoreSight component identity register 2 (MCUROM_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x1: ROM table component

0x0: Common identification value

MCU ROM CoreSight component identity register 3 (MCUROM_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

MCU ROM CoreSight component identity register 3 (MCUROM_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value
58.5.4 MCU ROM table register map

Table 704. MCU ROM table register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Function</th>
<th>Reset value</th>
<th>Offset</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFC0</td>
<td>MCUROMMEMTYPEPER</td>
<td></td>
<td></td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>0xFD0</td>
<td>MCUROM_PIDR4</td>
<td>SIZE [3:0]</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JEP106CON[3:0]</td>
<td></td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>0xFD4</td>
<td>MCUROM_PIDR0</td>
<td>PARTNUM[7:0]</td>
<td>X X X X X X</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>MCUROM_PIDR1</td>
<td>JEP106ID[3:0]</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PARTNUM[11:8]</td>
<td>X X X X X X</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>0xFE4</td>
<td>MCUROM_PIDR2</td>
<td>REV[0:3]</td>
<td>0</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PARTNUM[3:0]</td>
<td>X X X X X X</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>0xFE8</td>
<td>MCUROM_PIDR3</td>
<td>REVAND[3:0]</td>
<td>0</td>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CMOD[3:0]</td>
<td></td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>MCUROM_CIDR0</td>
<td>PREAMBLE[7:0]</td>
<td>0</td>
<td>56</td>
<td>0</td>
</tr>
<tr>
<td>0xFE4</td>
<td>MCUROM_CIDR1</td>
<td>CLASS[3:0]</td>
<td>0</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PREAMBLE[11:8]</td>
<td></td>
<td>64</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>68</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>72</td>
<td></td>
</tr>
</tbody>
</table>

Refer to Table 701: MCU ROM table for register boundary addresses.
58.5.5 Processor ROM table registers

CPU ROM memory type register (CPUROM_MEMTYPEPER)
Address offset: 0xFCC
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 SYSMEM: system memory
1: system memory present on this bus

CPU ROM CoreSight peripheral identity register 4 (CPUROM_PIDR4)
Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 SIZE[3:0]: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 JEP106CON[3:0]: JEP106 continuation code
0x4: ARM® JEDEC continuation code
### CPU ROM CoreSight peripheral identity register 0 (CPUROM_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 00C9

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
- 0xC9: Cortex®-M33

### CPU ROM CoreSight peripheral identity register 1 (CPUROM_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 00B4

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
- 0xB: ARM® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
- 0x4: Cortex®-M33
**CPU ROM CoreSight peripheral identity register 2 (CPUROM_PIDR2)**

Address offset: 0xFE8  
Reset value: 0x0000 000B

<table>
<thead>
<tr>
<th>Bit 31:8</th>
<th>Bit 7:4</th>
<th>Bit 3</th>
<th>Bit 2:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0: rev r0p0</td>
<td>1: Designer ID specified by JEDEC</td>
<td>0x3: Arm® JEDEC code</td>
<td></td>
</tr>
</tbody>
</table>

**CPU ROM CoreSight peripheral identity register 3 (CPUROM_PIDR3)**

Address offset: 0xFEC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:8</th>
<th>Bit 7:4</th>
<th>Bit 3:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>REVAND[3:0]: metal fix version</td>
<td>CMOD[3:0]: customer modified</td>
</tr>
<tr>
<td>0x0: No metal fix</td>
<td>0x0: no customer modifications</td>
<td></td>
</tr>
</tbody>
</table>
**CPU ROM CoreSight component identity register 0 (CPUROM_CIDR0)**

Address offset: 0xFF0
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[7:0]:** Component identification bits [7:0]
0x0D: Common identification value

---

**CPU ROM CoreSight peripheral identity register 1 (CPUROM_CIDR1)**

Address offset: 0xFF4
Reset value: 0x0000 0010

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **CLASS[3:0]:** Component identification bits [15:12] - component class
0x1: ROM table component

0x0: Common identification value
CPU ROM CoreSight component identity register 2 (CPUROM_CIDR2)
Address offset: 0xFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 PREAMBLE[19:12]: component identification bits [23:16]
0x05: common identification value

CPU ROM CoreSight component identity register 3 (CPUROM_CIDR3)
Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 PREAMBLE[27:20]: component identification bits [31:24]
0xB1: common identification value

58.5.6 Processor ROM table register map

Table 705. CPU ROM table register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>
Table 705. CPU ROM table register map and reset values (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFD0</td>
<td>CPUROM_PIDR4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFE0</td>
<td>CPUROM_PIDR0</td>
<td>1 1 0 0</td>
<td>1 0 0 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFE4</td>
<td>CPUROM_PIDR1</td>
<td>1 0 1 1</td>
<td>0 1 0 0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFE8</td>
<td>CPUROM_PIDR2</td>
<td>0 0 0 0</td>
<td>0 1 0 0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFE0</td>
<td>CPUROM_PIDR3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFF0</td>
<td>CPUROM_CIDR0</td>
<td>0 0 0 0</td>
<td>1 1 0 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFF4</td>
<td>CPUROM_CIDR1</td>
<td>0 0 0 0</td>
<td>1 0 0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFF8</td>
<td>CPUROM_CIDR2</td>
<td>0 0 0 0</td>
<td>0 1 0 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFFC</td>
<td>CPUROM_CIDR3</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Refer to Table 702: Processor ROM table for register boundary addresses.

58.6 Data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as one of the following:
- watchpoint
- ETM trigger
- PC sampling trigger
- data address sampling trigger
- data comparator (COMP 1 only)
- clock cycle counter comparator (COMP 0 only)

It also contains counters for:
- clock cycles
- folded instructions
- load store unit (LSU) operations
- sleep cycles
- number of cycles per instruction
- interrupt overhead
A DWT comparator compares the value held in its DWT comparator \textit{x} register (\textit{DWT_COMP}x\textit{R}) with one of the following:

- a data address
- an instruction address
- a data value
- the cycle-count value, for COMP 0 only

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- one or more DWT data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value
- a watchpoint debug event, on either the PC value or the accessed data address
- a CMPMATCH[N] event, that signals the match outside the DWT unit

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter debug state.

For more details on how to use the DWT, refer to the Arm®v8-M Architecture Reference Manual [4].

### 58.6.1 DWT registers

The DWT registers are located at address range 0xE000 1000 to 0xE000 1FFC.

#### DWT control register (DWT_CTRLR)

Address offset: 0x000

Reset value: 0x4000 0000

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Width</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMCOMP[3:0]</td>
<td>4</td>
<td>0x0000</td>
</tr>
<tr>
<td>NOSTRIPRKT</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>NODEXTTRG</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>NOFPGENT</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>NOPRECNT</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CYCDES</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CYCETENA</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FOLDETENA</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>LSUEETENA</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SLEEPEETENA</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>EXCEETENA</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>EXCTREGENA</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PCSAMPLENA</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SYNCTAP[1:0]</td>
<td>2</td>
<td>00</td>
</tr>
<tr>
<td>CYCTAP</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>POSTINIT[3:0]</td>
<td>4</td>
<td>0000</td>
</tr>
<tr>
<td>POSTRESET[3:0]</td>
<td>4</td>
<td>0000</td>
</tr>
<tr>
<td>CYCTREGENA</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Bits 31:28 **NUMCOMP[3:0]**: number of comparators implemented (read only)
- 0x4: four comparators

Bit 27 **NOTRCPKT**: trace sampling and exception tracing support (read only)
- 0: supported

Bit 26 **NOEXTRIG**: external match signal, CMPMATCH support (read only)
- 0: supported

Bit 25 **NOCYCNCNT**: cycle counter support (read only)
- 0: supported

Bit 24 **NOPRFCNT**: profiling counter support (read only)
- 0: supported

Bit 23 **CYCDISS**: cycle counter disabled secure.
- Controls whether the cycle counter is disabled in secure mode.
- 0: no effect
- 1: disable incrementing of the cycle counter when the processor is in secure state

Bit 22 **CYCEVTENA**: enable for POSTCNT underflow event counter packet generation
- 0: disabled
- 1: enabled

Bit 21 **FOLDEVTENA**: enable for folded instruction counter overflow event generation
- 0: disabled
- 1: enabled

Bit 20 **LSUEVTENA**: enable for LSU counter overflow event generation
- 0: disabled
- 1: enabled

Bit 19 **SLEEPEVTENA**: enable for sleep counter overflow event generation
- 0: disabled
- 1: enabled

Bit 18 **EXCEVTENA**: enable for exception overhead counter overflow event generation
- 0: disabled
- 1: enabled

Bit 17 **CPIEVTENA**: enable for CPI counter overflow event generation
- 0: disabled
- 1: enabled

Bit 16 **EXTRCEVA**: enable for exception trace generation
- 0: disabled
- 1: enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA**: enable for POSTCNT counter to be used as a timer for periodic PC sample packet generation
- 0: disabled
- 1: enabled
Bits 11:10 SYNCTAP[1:0]: position of the synchronization packet counter tap on the CYCCNT counter
   This field determines the synchronization packet rate.
   00: disabled, no synchronization packets
   01: Tap at CYCCNT[24]
   10: Tap at CYCCNT[26]
   11: Tap at CYCCNT[28]

Bit 9 CYCTAP: Selects the position of the POSTCNT tap on the CYCCNT counter.
   0: Tap at CYCCNT[6]
   1: Tap at CYCCNT[10]

Bits 8:5 POSTINIT[3:0]: initial value of the POSTCNT counter
   Writes to this field are ignored if POSTCNT counter is enabled. CYCEVTENA or PCSAMPLENA bits must be reset prior to writing POSTINIT.

Bits 4:1 POSTRESET[3:0]: reload value of the POSTCNT counter

Bit 0 CYCCNTENA: enable CYCCNT counter
   0: disabled
   1: enabled

**DWT cycle count register (DWT_CYCCNTR)**

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DWT CPI count register (DWT_CPICNTR)**

Address offset: 0x008
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*STI*
DWT exception count register (DWT_EXCCNT)

Address offset: 0x00C

Reset value: 0xXXXX XXXX

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXCCNT[7:0]**: exception overhead cycle counter
Counts the number of cycles spent in exception processing.

DWT sleep count register (DWT_SLPCNTR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SLEEP[7:0]**: sleep cycle counter
Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).
### DWT LSU count register (DWT_LSUCNTR)

Address offset: 0x014  
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

15 14 13 12 11 10  9 8 7 6 5 4 3 2 1 0  


Bits 31:8 Reserved, must be kept at reset value.  
Bits 7:0 **LSUCNT[7:0]**: load store counter  
Counts additional cycles required to execute load and store instructions.

### DWT fold count register (DWT_FOLDCNTR)

Address offset: 0x018  
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

15 14 13 12 11 10  9 8 7 6 5 4 3 2 1 0  


Bits 31:8 Reserved, must be kept at reset value.  
Bits 7:0 **FOLDCNT[7:0]**: folded instruction counter  
Increments on each instruction that takes 0 cycles.

### DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C  
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**EIASAMPLE[31:16]**

r r r r r r r r r r r r r  
15 14 13 12 11 10  9 8 7 6 5 4 3 2 1 0  

**EIASAMPLE[15:0]**

r r r r r r r r r r r r r  

2994/3119 RM0481 Rev 1
Bits 31:0  **EIASAMPLE[31:0]**: executed instruction address sample value.
Samples the current value of the program counter.

**DWT comparator x register (DWT_COMPxR)**
Address offset: 0x020 + 0x010 * x, (x = 0 to 3)
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td></td>
</tr>
</tbody>
</table>

**COMP[31:16]**

Bits 31:0  **COMP[31:0]**: reference value for comparison

**DWT function register 0 (DWT_FUNCTR0)**
Address offset: 0x028
Reset value: 0x5800 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

**ID[4:0]**
Identifies the capability for match for comparator 0.
0b01011: Cycle Counter, Instruction Address, Data Address and Data Address With Value

Bits 26:25  Reserved, must be kept at reset value.

Bit 24  **MATCHED**: comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12  Reserved, must be kept at reset value.
Bits 11:10 **DATAVSIZE[1:0]**: data value size
- Defines the size of the object being watched for by Data Value and Data Address comparators.
- \(0x0\): 1 byte
- \(0x1\): 2 bytes
- \(0x2\): 4 bytes
- \(0x3\): reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: action on match
- \(0x0\): trigger only
- \(0x1\): generate debug event
- \(0x2\): For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
- \(0x3\): For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: match type
- Controls the type of match generated by comparator 0.
- For possible values of this field, refer to [4].

**DWT function register 1 (DWT_FUNCTR1)**

Address offset: 0x038
Reset value: 0xD000 0000

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ID[4:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>MATCHED</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:27 **ID[4:0]**: capability identification
- Identifies the capability for match for comparator 1.
- \(0b11010\): Instruction Address, Instruction Address Limit, Data Address, Data Address Limit, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match
- Indicates if a comparator match has occurred since the register was last read.
- \(0\): no match
- \(1\): a match occurred

Bits 23:12 Reserved, must be kept at reset value.
Bits 11:10 **DATAVSIZE[1:0]**: data value size
Defines the size of the object being watched for by Data Value and Data Address comparators.
- 0x0: 1 byte
- 0x1: 2 bytes
- 0x2: 4 bytes
- 0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: action on match
- 0x0: trigger only
- 0x1: generate debug event
- 0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
- 0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: match type
Controls the type of match generated by comparator 1.
For possible values of this field, refer to [4].

### DWT function register 2 (DWT_FUNCTR2)
Address offset: 0x048
Reset value: 0x5000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for MATCH for comparator 2
- 0b01010: Instruction Address, Data Address, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match
Indicates if a comparator match has occurred since the register was last read.
- 0: no match
- 1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.
DWT function register 3 (DWT_FUNCTR3)

Address offset: 0x058

Reset value: 0xF000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER</td>
<td>USER</td>
<td>USER</td>
<td>USER</td>
<td>USER</td>
<td>USER</td>
<td>USER</td>
<td>USER</td>
<td>USER</td>
<td>USER</td>
<td>USER</td>
<td>USER</td>
<td>USER</td>
<td>USER</td>
<td>USER</td>
<td>USER</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:27 ID[4:0]: capability identification
Identifies the capability for MATCH for comparator 2.
0b11110: Instruction Address, Instruction Address Limit, Data Address, Data Address Limit,
Data value, Linked Data Value, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 MATCHED: comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.
Bits 11:10 **DATAVSIZE[1:0]:** data value size
- Defines the size of the object being watched for by Data Value and Data Address comparators.
- 0x0: 1 byte
- 0x1: 2 bytes
- 0x2: 4 bytes
- 0x3: reserved

Bits 9:6 Reserve, must be kept at reset value.

Bits 5:4 **ACTION[1:0]:** action on match
- 0x0: trigger only
- 0x1: Generate debug event
- 0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
- 0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]:** match type
- Controls the type of match generated by comparator 2.
- For possible values of this field, refer to [4]

### DWT device type architecture register (DWT_DEVARCHR)

Address offset: 0xFC8
Reset value: 0x4770 1A02

<table>
<thead>
<tr>
<th>Architect[10:0]</th>
<th>Present</th>
<th>Revision[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r   r   r   r   r   r   r   r   r   r   r   r   r   r   r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15   14   13   12   11   10   9   8   7   6   5   4   3   2   1   0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r   r   r   r   r   r   r   r   r   r   r   r   r   r   r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:21 **ARCHITECT[10:0]:** architect JEP106 code
- 0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm® limited.

Bit 20 **PRESENT:** DWT_DEVARCH register present
- 0x1: present

Bits 19:16 **REVISION[3:0]:** architecture revision
- 0x0: DWT architecture v2.0

Bits 15:12 **ARCHVER[3:0]:** architecture version
- 0x1: DWT architecture v2.0

Bits 11:0 **ARCHPART[11:0]:** architecture part
- 0xA02: DWT architecture
DWT device type register (DWT_DEVTYPEPER)

Address offset: 0xFCC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **Sub[3:0]**: sub-type
0x0: other

Bits 3:0 **Major[3:0]**: major type
0x0: miscellaneous

DWT CoreSight peripheral identity register 4 (DWT_PIDR4)

Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **Size[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106Con[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code
## DWT CoreSight peripheral identity register 0 (DWT_PIDR0)

Address offset: 0xFE0  
Reset value: 0x0000 0021

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.  

Bits 7:0  **PARTNUM[7:0]**: part number bits [7:0]  
0x21: Cortex®-M33 DWT part number

## DWT CoreSight peripheral identity register 1 (DWT_PIDR1)

Address offset: 0xFE4  
Reset value: 0x0000 00BD

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.  

Bits 7:4  **JEP106ID[3:0]**: JEP106 identity code bits [3:0]  
0xB: Arm® JEDEC code  

Bits 3:0  **PARTNUM[11:8]**: part number bits [11:8]  
0xD: Cortex®-M33 DWT part number
### DWT CoreSight peripheral identity register 2 (DWT_PIDR2)

Address offset: 0xFE8  
Reset value: 0x0000 000B

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>
```

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISON[3:0]**: component revision number  
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value  
0x1: designer identification specified by JEDEC

0x3: Arm® JEDEC code

### DWT CoreSight peripheral identity register 3 (DWT_PIDR3)

Address offset: 0xFEC  
Reset value: 0x0000 0000

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>
```

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version  
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified  
0x0: No customer modifications
DWT CoreSight component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** component identification bits [7:0]
- 0x0D: Common identification value

DWT CoreSight peripheral identity register 1 (DWT_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 0090

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** component identification bits [15:12] - component class
- 0x09: debug component

- 0x0: common identification value
DWT CoreSight component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 31| 30| 29| 28| 27| 26| 25| 24| 23| 22| 21| 20| 19| 18| 17| 16| 15| 14| 13| 12| 11| 10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
### Table 706. DWT register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset Register name</th>
<th>Reset value</th>
<th>Offset Register name</th>
<th>Reset value</th>
<th>Offset Register name</th>
<th>Reset value</th>
<th>Offset Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x004</td>
<td>DWT_CYCCNTR</td>
<td>CYCCNT[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>DWT_CPICNTR</td>
<td>CPICNT[7:0]</td>
<td>X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C</td>
<td>DWT_EXCCNTR</td>
<td>EXCCNT[7:0]</td>
<td>X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>DWT_SLPNCNTR</td>
<td>SLEEPCNT[7:0]</td>
<td>X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>DWT_LSUCNTR</td>
<td>LSUCNT[7:0]</td>
<td>X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x018</td>
<td>DWT_FOLDCNTR</td>
<td>FOLDCNT[7:0]</td>
<td>X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01C</td>
<td>DWT_PCSR</td>
<td>EIASAMPLE[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x020</td>
<td>DWT_COMP0R</td>
<td>COMP[31:0]</td>
<td>X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x024</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x02C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x030</td>
<td>DWT_COMP1R</td>
<td>COMP[31:0]</td>
<td>X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x034</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x03C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x040</td>
<td>DWT_COMP2R</td>
<td>COMP[31:0]</td>
<td>X X X X X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x044</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>Register name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
<td>---------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x050</td>
<td>DWT_COMP3R</td>
<td>COMP[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x054</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x058</td>
<td>DWT_FUNCTR3</td>
<td>ID[4:0] MATCHED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x05C</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFC8</td>
<td>DWT_DEVARCHR</td>
<td>ARCHITECT[10:0] PRESENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 1 0 0 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFC0</td>
<td>DWT_DEVTYPEPR</td>
<td>SUB[3:0] MAJOR[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD0</td>
<td>DWT_PIDR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD4</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>DWT_PIDR0</td>
<td>PARTNUM[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE4</td>
<td>DWT_PIDR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>1 0 1 1 1 1 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE8</td>
<td>DWT_PIDR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 1 0 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFEC</td>
<td>DWT_PIDR3</td>
<td>REVAND[3:0] CMOD[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF0</td>
<td>DWT_CIDR0</td>
<td>PREAMBLE[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 1 1 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 706. DWT register map and reset values (continued)
Refer to Table 702: Processor ROM table for register boundary addresses.

### 58.7 Instrumentation trace macrocell (ITM)

The ITM generates trace information in packets. Three sources can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The three sources in decreasing order of priority are the following:

- **Software trace**
  
  The software can write directly to any of 32 x 32-bit ITM stimulus registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access and the data written, into a packet that it writes to a FIFO. The ITM outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.

- **Hardware trace**
  
  The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.

- **Local timestamping**
  
  The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

For more information on the ITM and how to use it, refer to [4].

---

**Table 706. DWT register map and reset values (continued)**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Class[3:0]</th>
<th>Preamble[7:1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFF4</td>
<td>DWT_CIDR1</td>
<td></td>
<td>1 0 0 1 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0xFF8</td>
<td>DWT_CIDR2</td>
<td>0 0 0 0 0 1 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFFC</td>
<td>DWT_CIDR3</td>
<td>1 0 1 1 0 0 0 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Preamble[11:8]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFF4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFFC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
58.7.1 ITM registers

The ITM registers are located at address range 0xE000 0000 to 0xE000 0FFC.

**ITM stimulus register x (ITM_STIMRx)**

Address offset: 0x000 + 0x004 * x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

Condition: when writing

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**STIMULUS[31:16]**

Bits 31:0 **STIMULUS[31:0]**: trace output data

write data is output on the trace bus as a software event packet.

**ITM stimulus register x [alternate] (ITM_STIMRx)**

Address offset: 0x000 + 0x004 * x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

Condition: when reading

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:2 Reserved, must be kept at reset value.

**Bit 1 DISABLE**: Disable flag

0: stimulus port and ITM enabled
1: stimulus port and ITM disabled

**Bit 0 FIFO_READY**: FIFO ready indicator

0: stimulus port buffer is full (or port is disabled)
1: stimulus port can accept new write data
**ITM trace enable register (ITM_TER)**

Address offset: 0xE00

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**STIMENA[31:16]**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**STIMENA[15:0]**

Bits 31:0 **STIMENA[31:0]**: stimulus port enable

- Each bit xi (0 to 31) enables the stimulus port associated with the ITM_STIMRx register.
- 0: port disabled
- 1: port enabled

**ITM trace privilege register (ITM_TPR)**

Address offset: 0xE40

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**PRIVMASK[3:0]**

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRIVMASK[3:0]**: disable unprivileged access to ITM stimulus ports

- XXX0: unprivileged access permitted on ports 0 to 7
- XXX1: only privileged access permitted on ports 0 to 7
- X0XX: unprivileged access permitted on ports 8 to 15
- XX1X: only privileged access permitted on ports 8 to 15
- X0XX: unprivileged access permitted on ports 16 to 23
- XX1X: only privileged access permitted on ports 16 to 23
- XXX0: unprivileged access permitted on ports 24 to 31
- 1XXX: only privileged access permitted on ports 24 to 31
### ITM trace control register (ITM TCR)

Address offset: 0xE80  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:24 Reserved, must be kept at reset value.

- **Bit 23 BUSY**: indicates whether the ITM is currently processing events
  - 0: not busy
  - 1: busy

Bits 22:16 **TRACEBUSID[6:0]**: identifier for multi-source trace stream formatting  
If multi-source trace is in use, the debugger must write a non-zero value to this field.  

*Note*: Different identifiers must be used for each trace source in the system.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TSPRESCALE[1:0]**: local timestamp prescaler, used with the trace packet reference clock  
- 0x0: no prescaling
- 0x1: Divide by 4.
- 0x2: Divide by 16.
- 0x3: Divide by 64.

Bits 7:6 Reserved, must be kept at reset value.

- **Bit 5 STALENA**: stall enable
  - 0: Drop hardware source packets and generate an overflow if the ITM output is stalled.
  - 1: Stall the processor to guarantee delivery of data trace packets.

- **Bit 4 SWOENA**: SWO enable  
  Enables asynchronous clocking of the timestamp counter (read only).  
  0: Timestamp counter uses processor clock.

- **Bit 3 TXENA**: transmit enable  
  Enables forwarding of hardware event packets from the DWT unit to the trace port.  
  0: disabled
  1: enabled
Bit 2 **SYNCENA**: synchronization packet transmission enable
The debugger setting this bit must also configure the DWT_CTRLR.SYNCTAP field for the correct synchronization speed.
0: disabled
1: enabled

Bit 1 **TSENA**: local timestamp generation enable
0: disabled
1: enabled

Bit 0 **ITMENA**: ITM enable
0: disabled
1: enabled

**ITM device type architecture register (ITM_DEVARCHR)**

Address offset: 0xFBC

Reset value: 0x4770 1A01

```
+----------------+----------------+----------------+----------------+
|                |                |                |
| BIT 31:20      | BIT 19:12      | BIT 11:0       |
| ARCHITECT[10:0]| PRESENT        | ARCHPART[11:0] |
| ARCHITVER[3:0] |               | ARCHPART[11:0] |
| PRESENT        | REV1S[3:0]     |               |
| 15 14 13 12 11 | 10 9 8 7 6 5 4 | 3 2 1 0       |
```

Bits 31:21 **ARCHITECT[10:0]**: architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm® limited.

Bit 20 **PRESENT**: DEVARCH register presence
0x1: present

Bits 19:16 **REV1S[3:0]**: architecture revision
0x0: ITM architecture v2.0

Bits 15:12 **ARCHITVER[3:0]**: architecture version
0x1: ITM architecture v2.0

Bits 11:0 **ARCHITPART[11:0]**: architecture part
0xA01: ITM architecture
**ITM device type register (ITM_DEVTYPEPER)**

Address offset: 0xFCC  
Reset value: 0x0000 0043

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SUB[3:0]</th>
<th>MAJOR[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: sub-type  
0x4: associated with a bus, stimulus derived from bus activity

Bits 3:0 **MAJOR[3:0]**: major type  
0x3: trace source

**ITM CoreSight peripheral identity register 4 (ITM_PIDR4)**

Address offset: 0xFD0  
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SIZE[3:0]</th>
<th>JEP106CON[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size  
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code  
0x4: Arm® JEDEC code
**ITM CoreSight peripheral identity register 0 (ITM_PIDR0)**

Address offset: 0xFE0
Reset value: 0x0000 0021

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]

- 0x21: ITM part number

**ITM CoreSight peripheral identity register 1 (ITM_PIDR1)**

Address offset: 0xFE4
Reset value: 0x0000 00BD

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]

- 0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]

- 0xD: ITM part number
### ITM CoreSight peripheral identity register 2 (ITM_PIDR2)

Address offset: 0xFE8  
Reset value: 0x0000 000B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REVISION[3:0]**: component revision number  
0x0: r0p0

Bit 3  **JEDEC**: JEDEC assigned value  
0x1: designer identification specified by JEDEC

0x3: Arm® JEDEC code

### ITM CoreSight peripheral identity register 3 (ITM_PIDR3)

Address offset: 0xFEC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REVAND[3:0]**: metal fix version  
0x0: no metal fix

Bits 3:0  **CMOD[3:0]**: customer modified  
0x0: no customer modifications
**ITM CoreSight component identity register 0 (ITM_CIDR0)**

Address offset: 0xFF0

Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0


**Bits 31:8**  Reserved, must be kept at reset value.

**Bits 7:0**  **PREAMBLE[7:0]:** Component identification bits [7:0]

0x0D: Common identification value

**ITM CoreSight peripheral identity register 1 (ITM_CIDR1)**

Address offset: 0xFF4

Reset value: 0x0000 00E0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0


**Bits 31:8**  Reserved, must be kept at reset value.

**Bits 7:4**  **CLASS[3:0]:** Component identification bits [15:12] - component class

0xE: Trace generator component

**Bits 3:0**  **PREAMBLE[11:8]:** Component identification bits [11:8]

0x0: Common identification value
ITM CoreSight component identity register 2 (ITM_CIDR2)
Address offset: 0xFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Bits 31:8: Reserved, must be kept at reset value.
Bits 7:0: PREAMBLE[19:12]: Component identification bits [23:16]
0x05: Common identification value

ITM CoreSight component identity register 3 (ITM_CIDR3)
Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Bits 31:8: Reserved, must be kept at reset value.
Bits 7:0: PREAMBLE[27:20]: Component identification bits [31:24]
0xB1: Common identification value

58.7.2 ITM register map

The ITM registers are located at address range 0xE000 0000 to 0xE000 0FFC.

Table 707. ITM register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000 to 0x07C</td>
<td>ITM_STIMR0 to ITM_STIMR31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0x07C to 0x0DFC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xE00</td>
<td>ITM_TER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

3016/3119
Table 707. ITM register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE40</td>
<td>ITM_TPR</td>
<td>BUSY</td>
</tr>
<tr>
<td>0xE80</td>
<td>ITM_TCR</td>
<td>BUSY</td>
</tr>
<tr>
<td>0xFBC</td>
<td>ITM_DEVARCHR</td>
<td>ARCHITECT[10:0]</td>
</tr>
<tr>
<td>0xFD0</td>
<td>ITM_PIDR4</td>
<td>SIZE [3:0]</td>
</tr>
<tr>
<td>0xFD4</td>
<td>ITM_PIDR0</td>
<td>PARTNUM[7:0]</td>
</tr>
<tr>
<td>0xFD8</td>
<td>ITM_PIDR1</td>
<td>JEP106ID [3:0]</td>
</tr>
<tr>
<td>0xFD8</td>
<td>ITM_PIDR2</td>
<td>REVISION [3:0]</td>
</tr>
<tr>
<td>0xFF0</td>
<td>ITM_CIDR0</td>
<td>REVAND[3:0]</td>
</tr>
<tr>
<td>0xFF4</td>
<td>ITM_CIDR1</td>
<td>CLASS[3:0]</td>
</tr>
<tr>
<td>0xFFF</td>
<td>ITM_CIDR2</td>
<td>PREAMBLE[19:12]</td>
</tr>
<tr>
<td>0xFFC</td>
<td>ITM_CIDR3</td>
<td>PREAMBLE[27:20]</td>
</tr>
</tbody>
</table>
Refer to *Table 702: Processor ROM table* for register boundary addresses.

### 58.8 Breakpoint unit (BPU)

The BPU allows the user to set hardware breakpoints. It contains eight comparators that monitor the instruction fetch address. If a match occurs, the instruction comparators can be configured to generate a breakpoint instruction.

For more information on the breakpoint unit and how to use it, refer to [4].

#### 58.8.1 BPU registers

The BPU registers are located at address range 0xE0002000 to 0xE0002FFC.

**BPU control register (BPU_CTRLR)**

Address offset: 0x000  
Reset value: 0x1000 0080

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:28  **REV[3:0]**: revision number  
0x1: BPU version 2

Bits 27:15  Reserved, must be kept at reset value.

Bits 14:12, 7:4  **NUM_CODE[6:0]**: number of instruction address comparators supported  
0x08: 8 instruction comparators supported

Bits 11:8, 3:2  Reserved, must be kept at reset value.

Bit 1  **KEY**: Write protect key  
A write to FPB_CTRLR register is ignored if this bit is not set to 1.

Bit 0  **ENABLE**: FPB enable  
0: disabled  
1: enabled
### BPU comparator x register (BPU_COMPxR)

Address offset: 0x008 + 0x004 * x, (x = 0 to 7)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

BPADDR[31:16]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

BPADDR[15:1]

**Bits 31:1**  BPADDR[31:1]: breakpoint address

**Bit 0**  BE: breakpoint enable

0: disabled
1: enabled

### BPU device type architecture register (BPU_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4770 1A03

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
</tbody>
</table>

ARCHITECT[10:0]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
</tbody>
</table>

PRESEN

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
</tbody>
</table>

ARCHITECT[10:0]: architect JEP106 code

0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm® limited.

**Bit 20**  PRESEN: DEVARCH register present

0x1: present

**Bits 19:16**  REVISION[3:0]: architecture revision

0x0: BPU architecture v2.0

**Bits 15:12**  ARCHVER[3:0]: architecture version

0x1: BPU architecture v2.0

**Bits 11:0**  ARCHPART[11:0]: architecture part

0xA03: BPU architecture
## BPU device type register (BPU_DEVTYPEPER)

Address offset: 0xFCC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **SUB[3:0]**: sub-type
- 0x0: other

Bits 3:0  **MAJOR[3:0]**: major type
- 0x0: miscellaneous

## BPU CoreSight peripheral identity register 4 (BPU_PIDR4)

Address offset: 0xFD0  
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **SIZE[3:0]**: register file size
- 0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0  **JEP106CON[3:0]**: JEP106 continuation code
- 0x4: Arm® JEDEC code
### BPU CoreSight peripheral identity register 0 (BPU_PIDR0)

Address offset: 0xFE0  
Reset value: 0x0000 0021

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PARTNUM[7:0]**: part number bits [7:0]  
0x21: BPU part number

### BPU CoreSight peripheral identity register 1 (BPU_PIDR1)

Address offset: 0xFE4  
Reset value: 0x0000 00BD

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **JEP106ID[3:0]**: JEP106 identity code bits [3:0]  
0xB: Arm® JEDEC code

Bits 3:0  **PARTNUM[11:8]**: part number bits [11:8]  
0xD: BPU part number
BPU CoreSight peripheral identity register 2 (BPU_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 000B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REVISION[3:0]**: component revision number

0x0: r0p0

Bit 3  **JEDEC**: JEDEC assigned value

0x1: designer identification specified by JEDEC


0x3: Arm ® JEDEC code

BPU CoreSight peripheral identity register 3 (BPU_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REVAND[3:0]**: metal fix version

0x0: no metal fix

Bits 3:0  **CMOD[3:0]**: customer modified

0x0: no customer modifications
BPU CoreSight component identity register 0 (BPU_CIDR0)

Address offset: 0xFF0

Reset value: 0x00_00_00_0D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 \texttt{PREAMBLE}[7:0]: component identification bits [7:0]

0x0D: common identification value

BPU CoreSight peripheral identity register 1 (BPU_CIDR1)

Address offset: 0xFF4

Reset value: 0x00_00_00_90

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 \texttt{CLASS}[3:0]: component identification bits [15:12] - component class

0x09: debug component

Bits 3:0 \texttt{PREAMBLE}[11:8]: component identification bits [11:8]

0x0: common identification value
BPU CoreSight component identity register 2 (BPU_CIDR2)
Address offset: 0xFF8
Reset value: 0x0000 0005

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0   PREAMBLE[19:12]: component identification bits [23:16]
           0x05: common identification value

BPU CoreSight component identity register 3 (BPU_CIDR3)
Address offset: 0xFFC
Reset value: 0x0000 00B1

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0   PREAMBLE[27:20]: component identification bits [31:24]
           0xB1: common identification value

58.8.2   BPU register map
The BPU registers are located at address range 0xE000 2000 to 0xE000 2FFC.

Table 708. BPU register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>REV[3:0]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

58.8.2   BPU register map
The BPU registers are located at address range 0xE000 2000 to 0xE000 2FFC.

Table 708. BPU register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>REV[3:0]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Table 708. BPU register map and reset values (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0024</td>
<td>BPU_COMPTR to BPU_COMPTR</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x028</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFBC</td>
<td>BPU_DEVARCHR</td>
<td>0 1 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFC0</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFC4</td>
<td>BPU_DEVTYPE</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>SUB[3:0]</td>
<td>MAJ[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD0</td>
<td>BPU_PIDR4</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>SIZE[3:0]</td>
<td>JEP106CON[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD4</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>BPU_PIDR0</td>
<td>0 0 1 0 0 0 0 0 0 1</td>
<td>PARTNUM[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE4</td>
<td>BPU_PIDR1</td>
<td>1 0 1 1 1 1 1 0 1</td>
<td>JEP106ID[3:0]</td>
<td>PARTNUM[11:8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE8</td>
<td>BPU_PIDR2</td>
<td>0 0 0 0 0 1 0 1 1</td>
<td>REVISION[3:0]</td>
<td>JEP106ID[6:4]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFEC</td>
<td>BPU_PIDR3</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>REVAN[3:0]</td>
<td>CMOD[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF0</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF4</td>
<td>BPU_CIDR1</td>
<td>0 0 0 0 1 1 0 0 1</td>
<td>CLASS[3:0]</td>
<td>PREAMBLE[11:8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF8</td>
<td>BPU_CIDR2</td>
<td>1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>PREAMBLE[19:12]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFFC</td>
<td>BPU_CIDR3</td>
<td>1 0 1 1 1 0 0 0 0 0 0</td>
<td>PREAMBLE[27:20]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Table 702: Processor ROM table for register boundary addresses.

### 58.9 Embedded Trace Macrocell (ETM)

The ETM is a CoreSight™ component closely coupled to the CPU. The ETM generates trace packets that allow the execution of the Cortex®-M33 core to be traced. In the STM32H563/H573 and STM32H562, the ETM is configured for instruction trace only. Data accesses are not included in the trace information.
The ETM receives information from the CPU over the processor trace interface, including:

- number of instructions executed in the same cycle
- changes in program flow
- current processor instruction state
- addresses of memory locations accessed by load and store instructions
- type, direction and size of a transfer
- Condition code information
- exception information
- wait for interrupt state information

For more information, refer to the Arm® CoreSight™ ETM-M33 Technical Reference Manual [6].

58.9.1 ETM registers

The ETM registers are located at address range 0xE004 1000 to 0xE004 1FFC.

**ETM programming control register (ETM_PRGCTLR)**

Address offset: 0x004

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: trace unit enable

0: disabled
1: enabled
**ETM status register (ETM_STATR)**

Address offset: 0x00C

Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**PMSTABLE**: stability status

Indicates that the ETM-M33 registers are stable and can be read.

- 0: not stable
- 1: stable

**IDLE**: trace unit status

Indicates that the trace unit is inactive.

- 0: not idle
- 1: idle

**ETM configuration register (ETM_CONFIGR)**

Address offset: 0x010

Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**RS**: return stack enable

- 0: disabled
- 1: enabled

Bits 31:13 Reserved, must be kept at reset value.

Bits 31:13 Reserved, must be kept at reset value.
ETM event control 0 register (ETM_EVENTCTL0R)

Address offset: 0x020
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 15</td>
<td><strong>TYPE1</strong>: resource type for event1</td>
</tr>
<tr>
<td></td>
<td>0: single selected resource</td>
</tr>
<tr>
<td></td>
<td>1: boolean combined resource pair</td>
</tr>
<tr>
<td>Bits 14:12</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 11:8</td>
<td><strong>SEL1[3:0]</strong>: resource number based on TYPE1</td>
</tr>
<tr>
<td></td>
<td>Selects the resource number, based on the value of TYPE1.</td>
</tr>
<tr>
<td></td>
<td>When TYPE1 = 0, a single resource from 0-15 defined by SEL1[3:0] is selected.</td>
</tr>
<tr>
<td></td>
<td>When TYPE1 = 1, a boolean combined resource pair defined by SEL1[2:0] is selected.</td>
</tr>
<tr>
<td>Bit 7</td>
<td><strong>TYPE0</strong>: resource type for event0</td>
</tr>
<tr>
<td></td>
<td>0: single selected resource</td>
</tr>
<tr>
<td></td>
<td>1: boolean combined resource pair</td>
</tr>
<tr>
<td>Bits 6:4</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 3:0</td>
<td><strong>SEL0[3:0]</strong>: resource number based on TYPE0</td>
</tr>
<tr>
<td></td>
<td>Selects the resource number, based on the value of TYPE0.</td>
</tr>
<tr>
<td></td>
<td>When TYPE0 = 0, a single resource from 0-15 defined by SEL0[3:0] is selected.</td>
</tr>
<tr>
<td></td>
<td>When TYPE0 = 1, a boolean combined resource pair defined by SEL0[2:0] is selected.</td>
</tr>
</tbody>
</table>
ETM event control 1 register (ETM_EVENTCTL1R)
Address offset: 0x024
Reset value: 0xXXXX XXXX

```
+---------+---------+---------+---------+
<p>| 31      | 30      | 29      | 28      |
| Reserved | Reserved | Reserved | Reserved |
| 27      | 26      | 25      | 24      |
| Reserved | Reserved | Reserved | Reserved |
| 23      | 22      | 21      | 20      |
| Reserved | Reserved | Reserved | Reserved |
| 19      | 18      | 17      | 16      |</p>
<table>
<thead>
<tr>
<th>Reserved</th>
<th>Reserved</th>
<th>Reserved</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
```

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LPOVERRIDE**: low-power state behavior override
- 0: normal low-power state behavior
- 1: The resources and event trace generation are not affected by entry to a low-power state.

Bit 11 **ATB**: ATB trigger enable
- 0: disabled
- 1: enabled

Bits 10:2 Reserved, must be kept at reset value.

Bits 1:0 **INSTEN[1:0]**: instruction event generation
- Enables generation of an event element in the instruction stream.
  - 0bX0: Event0 does not cause an event element.
  - 0bX1: Event0 causes an event element when it occurs.
  - 0b0X: Event1 does not cause an event element.
  - 0b1X: Event1 causes an event element when it occurs.

ETM stall control register (ETM_STALLCTRLR)
Address offset: 0x02C
Reset value: 0xXXXX XXXX

```
+---------+---------+---------+---------+
<p>| 31      | 30      | 29      | 28      |
| Reserved | Reserved | Reserved | Reserved |
| 27      | 26      | 25      | 24      |
| Reserved | Reserved | Reserved | Reserved |
| 23      | 22      | 21      | 20      |
| Reserved | Reserved | Reserved | Reserved |
| 19      | 18      | 17      | 16      |</p>
<table>
<thead>
<tr>
<th>Reserved</th>
<th>Reserved</th>
<th>Reserved</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
```

```
+---------+---------+---------+---------+---------+---------+
<p>| 31      | 30      | 29      | 28      | 27      | 26      |
| Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| 25      | 24      | 23      | 22      | 21      | 20      |
| Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| 19      | 18      | 17      | 16      | Reserved | Reserved |</p>
<table>
<thead>
<tr>
<th>Reserved</th>
<th>Reserved</th>
<th>Reserved</th>
<th>Reserved</th>
<th>Reserved</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
```

- **INSTPRIORITY**:
- **ISTALL**: Level [3:0]
- **LEVEL[3:0]**

**Note:**
- **LPOVERRIDE**: low-power state behavior override
- **ATB**: ATB trigger enable
- **INSTEN[1:0]**: instruction event generation
- **INSTPRIORITY**: instruction priority
- **ISTALL**: instruction stall
- **LEVEL[3:0]**: level [3:0]
Bits 31:11  Reserved, must be kept at reset value.

Bit 10  **INSTPRIORITY**: instruction trace priority
Prioritizes instruction trace if instruction trace buffer space is less than LEVEL[3:0].
0: The ETM must not prioritize instruction trace.
1: The ETM can prioritize instruction trace.

Bit 9  Reserved, must be kept at reset value.

Bit 8  **INSTALL**: processor stalling
Stalls processor based on instruction trace buffer space.
0: The ETM must not stall the processor.
1: The ETM can stall the processor.

Bits 7:4  Reserved, must be kept at reset value.

Bits 3:0  **LEVEL[3:0]**: Threshold at which stalling becomes active
This field provides four levels. This level can be varied to optimize the level of invasion
caused by stalling, balanced against the risk of a FIFO overflow.
0x0: zero invasion, but greater risk of FIFO overflow
... 0xF: maximum invasion but less risk of FIFO overflow

**ETM synchronization period register (ETM_SYNCPR)**
Address offset: 0x034
Reset value: 0x0000 000A

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| PERIOD[4:0] |
| r | r | r | r | r |

Bits 31:5  Reserved, must be kept at reset value.

Bits 4:0  **PERIOD[4:0]**: synchronization period
Defines the number of bytes of trace between trace synchronization requests as a total of the
number of bytes generated by the instruction stream.
0xA: 1024 bytes
**ETM cycle count control register (ETM_CCCTL)R**

Address offset: 0x038  
Reset value: 0xXXXX XXXX

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |   |   |   |   |  

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0  
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  

**Bits 31:12** Reserved, must be kept at reset value.  
**Bits 11:0** **THRESHOLD[11:0]:** instruction trace cycle count threshold  
Sets the threshold value for instruction trace cycle counting. The threshold represents the minimum interval between cycle-count trace packets.

**ETM trace identification register (ETM_TRACEIDR)**  
Address offset: 0x040  
Reset value: 0xXXXX XXXX

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |   |   |   |   |  

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0  
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  

**Bits 31:7** Reserved, must be kept at reset value.  
**Bits 6:0** **TRACEID[6:0]:** Trace identification to output onto the trace bus  
This field must be programmed with a unique value to differentiate it from other trace sources in the system.  
Values 0x00 and 0x70-0x7F are reserved.

**ETM ViewInst main control register (ETM_VICTLR)**  
Address offset: 0x080  
Reset value: 0xXXXX XXXX

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |   |   |   |   |  

**EXLEVEL_S[3:0]**
ETM counter reload value register 0 (ETM_CNTRLDVR0)

Address offset: 0x140
Reset value: 0xFFFF XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VALUE[15:0]**: counter reload value
This value is loaded into the counter each time the reload event occurs.
ETM identification register 8 (ETM_IDR8)
Address offset: 0x180
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**MAXSPEC[31:16]**

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16

MAXSPEC[15:0]

Bits 31:0 **MAXSPEC[31:0]**: maximum speculation depth
Indicates the maximum speculation depth of the instruction trace stream. This is the maximum number of P0 elements that have not been committed in the trace stream at any one time.
0x0: The maximum trace speculation depth is zero.

ETM identification register 9 (ETM_IDR9)
Address offset: 0x184
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**NUMP0KEY[31:16]**

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16

NUMP0KEY[15:0]

Bits 31:0 **NUMP0KEY[31:0]**: number of P0 right-hand keys used
0x0: no P0 right-hand keys used in instruction trace

ETM identification register 10 (ETM_IDR10)
Address offset: 0x188
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**NUMP1KEY[31:16]**

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16

NUMP1KEY[15:0]
ETM identification register 11 (ETM_IDR11)
Address offset: 0x18C
Reset value: 0x0000 0000

Bits 31:0 NUMP1KEY[31:0]: number of P1 right-hand keys used (including normal and special keys)
0x0: no P1 right-hand keys used in instruction trace

ETM identification register 12 (ETM_IDR12)
Address offset: 0x190
Reset value: 0x0000 0001

Bits 31:0 NUMP1SPC[31:0]: number of special P1 right-hand keys used
0x0: no special P1 right-hand keys used in any configuration

ETM identification register 13 (ETM_IDR13)
Address offset: 0x194
Reset value: 0x0000 0000

Bits 31:0 NUMCONDKEY[31:0]: number of conditional instruction right-hand keys used (including normal and special keys)
0x1: one conditional instruction right-hand key implemented

Bits 31:0 NUMCONDSPC[31:0]: number of special conditional instruction right-hand keys used
Bits 31:0 **NUMCONDSPC[31:0]**: number of special conditional instruction right-hand keys used

0x0: no special conditional instruction right-hand keys implemented

**ETM implementation specific register 0 (ETM_IMSPECR0)**

Address offset: 0x1C0
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SUPPORT[3:0]**: implementation specific extension support

0x0: no implementation specific extensions are supported

**ETM identification register 0 (ETM_IDR0)**

Address offset: 0x1E0
Reset value: 0x2800 06E1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **COMMOPT**: commit field meaning

Indicates the meaning of the commit field in some packets.

1: commit mode 1

Bits 28:18 Reserved, must be kept at reset value.

Bit 17 **TRCEXDATA**: trace data transfers for exceptions

Indicates support for the tracing of data transfers for exceptions and exception returns.

0: not implemented

Bits 16:15 **QSUPP[1:0]**: Q element support

0: not supported
ETM identification register 1 (ETM_IDR1)

Address offset: 0x1E4
Reset value: 0x4100 F421

Bits 31:24 **DESIGNER[7:0]**: trace unit designer
0x41: Arm®

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:8 **TRCARCHMAJ[3:0]**: major trace unit architecture version number
0x4: ETMv4

Bits 7:4 **TRCARCHMIN[3:0]**: minor trace unit architecture version number
0x2: minor revision 2

Bits 3:0 **REVISION[3:0]**: implementation revision number
0x1: implementation revision 1

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **CONDTYPE[1:0]**: conditional results tracing
Indicates how conditional results are traced.
0: The trace unit indicates only if a conditional instruction passes or fails its condition code check

Bits 11:10 **NUMEVENT[1:0]**: Number of events supported
0x1: two events

Bit 9 **RESTACK**: return stack support
1: two entry return stacks

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TRCCCI**: cycle counting support
1: cycle counting implemented

Bit 6 **TRCCOND**: conditional instruction support
1: conditional instruction tracing implemented

Bit 5 **TRCBB**: branch broadcast support
1: branch broadcast tracing implemented

Bits 4:3 **TRCDATA[1:0]**: data tracing support
0x0: data tracing not supported

Bits 2:1 **INSTP0[1:0]**: support for tracing of load and store instructions as P0 elements
0x0: not supported

Bit 0 Reserved, must be kept at reset value.
ETM identification register 2 (ETM_IDR2)

Address offset: 0x1E8
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td>r</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>r</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DASIZE[0]</th>
<th>VMIDSIZE[4:0]</th>
<th>CIDSIZE[4:0]</th>
<th>IASIZE[4:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **CCSIZE[3:0]**: cycle counter size
0x0: 12 bits

Bits 24:20 **DVSIZE[4:0]**: data value size
0x0: data value size not supported

Bits 19:15 **DASIZE[4:0]**: data address size.
0x0: data address size not supported

Bits 14:10 **VMIDSIZE[4:0]**: virtual machine ID size
0x0: virtual machine ID tracing not implemented

Bits 9:5 **CIDSIZE[4:0]**: context ID size
0x0: context ID tracing not implemented

Bits 4:0 **IASIZE[4:0]**: instruction address size
0x4: maximum 32-bit address size

ETM identification register 3 (ETM_IDR3)

Address offset: 0x1EC
Reset value: 0x0F09 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>r</td>
<td>f</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>r</td>
<td>f</td>
<td>r</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **NUMPROC[2:0]**: number of processors
0x0: 2 processors

Bits 24:20 **SYSTAL**: system stall
0x0: system stall supported

Bits 23:15 **STALLCL**: stall count
0x0: stall count not supported

Bits 14:10 **SYNC/PR**: sync/processor synchronisation
0x0: sync/processor synchronisation not supported

Bits 9:5 **TRCERR**: tracing error
0x0: tracing error not supported

Bits 4:0 **EXLEVEL_S[3:0]**: exception level
0x3: exception level 3
Bit 31  **NOOVERFLOW**: ETM_STALLCTRL.NOOVERFLOW implementation
0: not implemented

Bits 30:28  **NUMPROC[2:0]**: number of processors available for tracing
0x0: one processor

Bit 27  **SYSSTALL**: system support for stall control of the processor
1: system supports stall control

Bit 26  **STALLCTL**: stall control support
1: ETM_STALLCTRL implemented

Bit 25  **SYNCR**: trace synchronization period support
1: ETM_SYNCR is read-only for instruction trace only configuration. The trace synchronization period is fixed.

Bit 24  **TRCERR**: ETM_VICTLR.TRCErr implementation
0x1: implemented

Bits 23:20  Reserved, must be kept at reset value.

Bits 19:16  **EXLEVEL_S[3:0]**: privilege levels implementation
0x9: privilege levels thread and handler implemented

Bits 15:12  Reserved, must be kept at reset value.

Bits 11:0  **CCITMIN[11:0]**: minimum value that can be programmed to TRCCCTLR.THRESHOLD
0x4: minimum of four-instruction trace cycles

**ETM identification register 4 (ETM_IDR4)**

Address offset: 0x1F0
Reset value: 0x0011 4000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>SUPPCD</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:28  **NUMVMIDC[3:0]**: number of virtual machine ID (VMID) comparators
0x0: VMID comparators not implemented

Bits 27:24  **NUMCIDC[3:0]**: number of context ID comparators
0x0: context ID comparators not supported

Bits 23:20  **NUMSSCC[3:0]**: number of single-shot comparator controls
0x1: one single-shot comparator control implemented

Bits 19:16  **NUMRSPAIR[3:0]**: number of resource selection pairs
0x1: two resource selection pairs implemented
Bits 15:12  **NUMPC[3:0]**: number of processor comparator inputs for the DWT
   0x4: four processor comparator inputs implemented

Bits 11:9  Reserved, must be kept at reset value.

   Bit 8  **SUPPDAC**: data address comparisons
       0: data address comparisons not supported

Bits 7:4  **NUMDVC[3:0]**: number of data value comparators
       0x0: no data value comparators implemented

Bits 3:0  **NUMACPAIRS[3:0]**: number of address comparator pairs
       0x0: no address comparator pairs implemented

**ETM identification register 5 (ETM_IDR5)**

Address offset: 0x1F4

Reset value: 0x90C7 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31  **REDFUNCNTR**: reduced function counter
       1: counter 0 implemented as a reduced function counter

Bits 30:28  **NUMCNTR[2:0]**: number of counters
       0x1: one counter implemented.

Bits 27:25  **NUMSEQSTATE[2:0]**: number of sequencer states
       0x0: no sequencer states implemented.

   Bit 24  Reserved, must be kept at reset value.

   Bit 23  **LPOVERRIDE**: low-power state override support
       1: low-power state override support implemented

   Bit 22  **ATBTRIG**: ATB trigger support
       1: ATB trigger support implemented

Bits 21:16  **TRACEIDSIZE[5:0]**: number of bits of trace identification
       0x7: 7-bit trace identification implemented

Bits 15:12  Reserved, must be kept at reset value.

Bits 11:9  **NUMEXTINSEL[2:0]**: number of external input selectors
       0x0: no external input selectors implemented.

   Bits 8:0  **NUMEXTIN[8:0]**: number of external inputs
       0x004: four external inputs implemented.
### ETM resource register 2 (ETM_RSCTRLR2)

Address offset: 0x208  
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>Bits 31:22</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 21</td>
<td><strong>PAIRINV</strong>: result of a combined pair of resources inversion</td>
</tr>
<tr>
<td></td>
<td>0: not inverted</td>
</tr>
<tr>
<td></td>
<td>1: inverted</td>
</tr>
<tr>
<td>Bit 20</td>
<td><strong>INV</strong>: selected resources inversion</td>
</tr>
<tr>
<td></td>
<td>0: not inverted</td>
</tr>
<tr>
<td></td>
<td>1: inverted</td>
</tr>
<tr>
<td>Bit 19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 18:16</td>
<td><strong>GROUP[2:0]</strong>: group of resources selection</td>
</tr>
<tr>
<td></td>
<td>0x0: external input selectors (select 0-3)</td>
</tr>
<tr>
<td></td>
<td>0x1: inputs from processor DWT comparators element (select 0-3)</td>
</tr>
<tr>
<td></td>
<td>0x2: counter at zero (select 0)</td>
</tr>
<tr>
<td></td>
<td>0x3: single-shot comparator (select 0)</td>
</tr>
<tr>
<td></td>
<td>Others: reserved</td>
</tr>
<tr>
<td>Bits 15:8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 7:0</td>
<td><strong>SELECT[7:0]</strong>: more resources selection</td>
</tr>
<tr>
<td></td>
<td>Selects one or more resources from the group selected in <strong>GROUP[2:0]</strong>.</td>
</tr>
</tbody>
</table>

### ETM resource register 3 (ETM_RSCTRLR3)

Address offset: 0x20C  
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>Bits 31:22</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 21</td>
<td><strong>PAIRINV</strong>: result of a combined pair of resources inversion</td>
</tr>
<tr>
<td></td>
<td>0: not inverted</td>
</tr>
<tr>
<td></td>
<td>1: inverted</td>
</tr>
<tr>
<td>Bit 20</td>
<td><strong>INV</strong>: selected resources inversion</td>
</tr>
<tr>
<td></td>
<td>0: not inverted</td>
</tr>
<tr>
<td></td>
<td>1: inverted</td>
</tr>
<tr>
<td>Bit 19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

| Bits 18:16 | **GROUP[2:0]**: group of resources selection  |
|            | 0x0: external input selectors (select 0-3)  |
|            | 0x1: inputs from processor DWT comparators element (select 0-3)  |
|            | 0x2: counter at zero (select 0)  |
|            | 0x3: single-shot comparator (select 0)  |
|            | Others: reserved  |
| Bits 7:0   | **SELECT[7:0]**: more resources selection  |
|            | Selects one or more resources from the group selected in **GROUP[2:0]**. |
ETM single-shot comparator control register 0 (ETM_SSCCR0)

Address offset: 0x280

Reset value: 0xXXXX XXXX

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **INV**: selected resources inversion
- 0: not inverted
- 1: inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: group of resources selection
- 0x0: external input selectors (select 0-3)
- 0x1: inputs from processor DWT comparators element (select 0-3)
- 0x2: counter at zero (select 0)
- 0x3: single-shot comparator (select 0)
- Others: reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: more resources selection
Selects one or more resources from the group selected in GROUP[2:0].

Bit 24 **RST**: single-shot comparator reset
Enables the single-shot comparator resource to be reset when it occurs, to enable another comparator match to be detected.
- 1: reset enabled

Bits 23:0 Reserved, must be kept at reset value.
### ETM single-shot comparator status register 0 (ETM_SSCSR0)

Address offset: 0xA0

Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>STATUS: single-shot comparator status</td>
<td>1</td>
<td>Indicates whether any of the selected comparators have matched.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0: no match occurred</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1: at least one match occurred</td>
</tr>
<tr>
<td>30:4</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>PC: processor comparator input sensitivity</td>
<td>1</td>
<td>1: single-shot comparator sensitive to processor comparator inputs</td>
</tr>
<tr>
<td>2</td>
<td>DV: data value comparator support</td>
<td>0</td>
<td>0: single-shot data value comparisons not supported</td>
</tr>
<tr>
<td>1</td>
<td>DA: data address comparator support</td>
<td>0</td>
<td>0: single-shot data address comparisons not supported</td>
</tr>
<tr>
<td>0</td>
<td>INST: instruction address comparator support</td>
<td>0</td>
<td>0: single-shot instruction address comparisons not supported</td>
</tr>
</tbody>
</table>

### ETM single-shot processor comparator input control register 0 (ETM_SSPCICR0)

Address offset: 0x20

Reset value: 0x20000000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>PC[3:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **PC[3:0]**: processor comparator inputs selection for single-shot control
- 0xXXX0: processor comparator input 0 not selected
- 0xXXX1: processor comparator input 0 selected
- 0xxX0X: Processor comparator input 1 not selected
- 0XX1X: processor comparator input 1 selected
- 0xX0XX: processor comparator input 2 not selected
- 0XX1XX: processor comparator input 2 selected
- 0x0XXX: processor comparator input 3 not selected
- 0x1XXX: processor comparator input 3 selected

**ETM power-down control register (ETM_PDCR)**

Address offset: 0x310

Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RW</td>
</tr>
</tbody>
</table>

Bits 31:4  Reserved, must be kept at reset value.

Bit 3  **PU**: power-up request
- 0: power-up not requested
- 1: power-up requested

Bits 2:0  Reserved, must be kept at reset value.

**ETM power-down status register (ETM_PDSR)**

Address offset: 0x314

Reset value: 0x0000 0003

<p>| | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

<p>| | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POWER</td>
</tr>
</tbody>
</table>

STICKYPD
Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **STICKYPD**: sticky power-down state
0: Trace register power has not been removed since the ETM_PDSR was last read.
1: Trace register power has been removed since the ETM_PDSR was last read.

Bit 0 **POWER**: ETM power-up status
1: ETM powered up

**ETM claim tag set register (ETMCLAIMSETR)**

Address offset: 0xFA0
Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

CLAIMSET[3:0] rw rw rw rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: claim tag bits setting
Write:
0000: no effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

**ETM claim tag clear register (ETMCLAIMCLR)**

Address offset: 0xFA4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

CLAIMCLR[3:0] rw rw rw rw
Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **CLAIMCLR[3:0]:** claim tag bits reset

Write:
- 0000: no effect
- xxx1: Clears bit 0.
- xx1x: Clears bit 1.
- x1xx: Clears bit 2.
- 1xxx: Clears bit 3.

Read: Returns current value of claim tag.

### ETM authentication status register (ETM_AUTHSTATR)

Address offset: 0xFB8
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:6  **SNID[1:0]:** security level for secure non-invasive debug
- 0x2: secure non-invasive debug disabled
- 0x3: secure non-invasive debug enabled

Bits 5:4  **SID[1:0]:** security level for secure invasive debug
- 0x0: not implemented

Bits 3:2  **NSNID[1:0]:** security level for non-secure non-invasive debug
- 0x2: non-secure non-invasive debug disabled
- 0x3: non-secure non-invasive debug enabled

Bits 1:0  **NSID[1:0]:** security level for non-secure invasive debug
- 0x0: not implemented
ETM device type architecture register (ETM_DEVARCHR)

Address offset: 0xFBC
Reset value: 0x4772 4A13

<table>
<thead>
<tr>
<th></th>
<th>ARCHITECT[10:0]</th>
<th>PRESEN</th>
<th>REVISION[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:21</td>
<td>ARCHITECT[10:0]: architect JEP106 code</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm® limited.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 20</td>
<td>PRESENT: DEVARCH register presence</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1: present</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 19:16</td>
<td>REVISION[3:0]: architecture revision</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x2: ETM architecture v4.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 15:12</td>
<td>ARCHVER[3:0]: architecture version</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4: ETM architecture v4.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 11:0</td>
<td>ARCHPART[11:0]: architecture part</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0xA13: ETM architecture</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ETM CoreSight device type register (ETM_DEVTYPEPER)

Address offset: 0xFCC
Reset value: 0x0000 0013

<table>
<thead>
<tr>
<th></th>
<th>SUBTYPE[3:0]</th>
<th>MAJORTYPE[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:8</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>Bits 7:4</td>
<td>SUBTYPE[3:0]: device sub-type identifier</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1: processor trace</td>
<td></td>
</tr>
<tr>
<td>Bits 3:0</td>
<td>MAJORTYPE[3:0]: device main type identifier</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x3: trace source</td>
<td></td>
</tr>
</tbody>
</table>
### ETM CoreSight peripheral identity register 4 (ETM_PIDR4)

Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SIZE[3:0]</td>
<td>JEP106CON[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:8** Reserved, must be kept at reset value.

**Bits 7:4** **SIZE[3:0]:** register file size

0x0: The register file occupies a single 4-Kbyte region.

**Bits 3:0** **JEP106CON[3:0]:** JEP106 continuation code

0x4: Arm® JEDEC code

### ETM CoreSight peripheral identity register 0 (ETM_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 0021

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PARTNUM[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:8** Reserved, must be kept at reset value.

**Bits 7:0** **PARTNUM[7:0]:** part number bits [7:0]

0x21: ETM part number
**ETM CoreSight peripheral identity register 1 (ETM_PIDR1)**

Address offset: 0xFE4  
Reset value: 0x0000 00BD

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **JEP106ID[3:0]**: JEP106 identity code bits [3:0]

- 0xB: Arm® JEDEC code

Bits 3:0  **PARTNUM[11:8]**: part number bits [11:8]

- 0xD: ETM part number

**ETM CoreSight peripheral identity register 2 (ETM_PIDR2)**

Address offset: 0xFE8  
Reset value: 0x0000 001B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REVISION[3:0]**: component revision number

- 0x1: r0p1

Bit 3  **JEDEC**: JEDEC assigned value

- 0x1: designer identification specified by JEDEC


- 0x3: Arm® JEDEC code
## ETM CoreSight peripheral identity register 3 (ETM_PIDR3)

Address offset: 0xFEC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.  
Bits 7:4  **REVAND[3:0]**: metal fix version  
0x0: no metal fix  
Bits 3:0  **CMOD[3:0]**: customer modified  
0x0: no customer modifications

## ETM CoreSight component identity register 0 (ETM_CIDR0)

Address offset: 0xFF0  
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:8  Reserved, must be kept at reset value.  
Bits 7:0  **PREAMBLE[7:0]**: component identification bits [7:0]  
0x0D: common identification value
ETM CoreSight peripheral identity register 1 (ETM_CIDR1)
Address offset: 0xFF4
Reset value: 0x0000 0090

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:4</td>
<td><strong>CLASS[3:0]</strong>: component identification bits [15:12] - component class</td>
</tr>
<tr>
<td></td>
<td>0x9: trace generator component</td>
</tr>
<tr>
<td>Bits 3:0</td>
<td><strong>PREAMBLE[11:8]</strong>: component identification bits [11:8]</td>
</tr>
<tr>
<td></td>
<td>0x0: common identification value</td>
</tr>
</tbody>
</table>

ETM CoreSight component identity register 2 (ETM_CIDR2)
Address offset: 0xFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td><strong>PREAMBLE[19:12]</strong>: component identification bits [23:16]</td>
</tr>
<tr>
<td></td>
<td>0x05: common identification value</td>
</tr>
</tbody>
</table>
ETM CoreSight component identity register 3 (ETM_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits 31:8</th>
<th>Bits 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x004</td>
<td>ETM_PRGCTLR</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x008</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x00C</td>
<td>ETM_STATR</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x010</td>
<td>ETM_CONFIGR</td>
<td>X XX</td>
<td>IDLE</td>
</tr>
<tr>
<td>0x014 to 0x01C</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x020</td>
<td>ETM_EVENTCTL0R</td>
<td>X XX</td>
<td>INSTEn[1:0]</td>
</tr>
<tr>
<td>0x024</td>
<td>ETM_EVENTCTL1R</td>
<td>X X X X X</td>
<td>INSTEn[1:0]</td>
</tr>
<tr>
<td>0x028</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

58.9.2 ETM register map

The ETM registers are accessed by the debugger at address range 0xE0041000 to 0xE0041FFC.
## Table 709. ETM register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02C</td>
<td>ETM_STALLCTRLR</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0x030</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x034</td>
<td>ETM_SYNCPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x038</td>
<td>ETM_CCCTLR</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0x03C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x040</td>
<td>ETM_TRACEIDR</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0x044 to 0x07C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x080</td>
<td>ETM_VICTLR</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0x084 to 0x0c3</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x140</td>
<td>ETM_CNTRLDVR0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0x144 to 0x17C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x180</td>
<td>ETM_IDR8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x184</td>
<td>ETM_IDR9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x188</td>
<td>ETM_IDR10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x18c</td>
<td>ETM_IDR11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x190</td>
<td>ETM_IDR12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x194</td>
<td>ETM_IDR13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x198 to 0x1BC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C0</td>
<td>ETM_IMSPECRC0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C4 to 0x1DC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 709. ETM register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1E0</td>
<td>ETM_IDR0</td>
<td>COMMOPT</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRC/EXDATA</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GR/PP[1:0]</td>
<td>0 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COND/TYPE [1:0]</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RET/STACK</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRCCOND</td>
<td>1 1 1 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRCCB</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRC/TPA[1:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INST/PR[1:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x1E4</td>
<td>ETM_IDR1</td>
<td>DESIGNER[7:0]</td>
<td>0 1 0 0 0 0 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRCARCHMI[3:0]</td>
<td>0 1 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRCARCHM[3:0]</td>
<td>1 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REVISION[3:0]</td>
<td>1 0 0 0 1</td>
</tr>
<tr>
<td>0x1E8</td>
<td>ETM_IDR2</td>
<td>CC SIZE[3:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DV SIZE[4:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DASIZE[4:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VMIDSIZE[4:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CIDSIZE[4:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IASIZE[4:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0x1EC</td>
<td>ETM_IDR3</td>
<td>NO OVERFLOW</td>
<td>0 0 0 0 0 1 1 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRC/EXDATA</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GTL/LEVEL_S[3:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CC ITMIN[11:0]</td>
<td>1 0 0 0 0</td>
</tr>
<tr>
<td>0x1F0</td>
<td>ETM_IDR4</td>
<td>NUM/VMDIC [3:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUM/CIDC [3:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUM/SSC[3:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUM/RSPAIR[3:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUM/PC[3:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUPP/DAC</td>
<td>0 1 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUM/VC[3:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUM/PAIRS[3:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0x1F4</td>
<td>ETM_IDR5</td>
<td>RED/FUNC/ST[2:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUM/CTRL[2:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUM/SSC[8:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPOVERRIDE</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRACE/IDSIZE[5:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUM/EXT/SEL[8:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUM/EXT/IN[8:0]</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0x1F8 to 0x204</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x208</td>
<td>ETM_RSCCTR2</td>
<td>PAIR/INV</td>
<td>X X X X X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GROUP[2:0]</td>
<td>X X X X X X X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SELECT[7:0]</td>
<td>X X X X X X X</td>
</tr>
<tr>
<td>0x20C</td>
<td>ETM_RSCCTR3</td>
<td>INV</td>
<td>X X X X X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GROUP[2:0]</td>
<td>X X X X X X X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SELECT[7:0]</td>
<td>X X X X X X X</td>
</tr>
<tr>
<td>0x210 to 0x27C</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x280</td>
<td>ETM_SCCCR0</td>
<td>RST</td>
<td>X</td>
</tr>
<tr>
<td>0x284 to 0x29C</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x2A0</td>
<td>ETM_SCCSR0</td>
<td>PC</td>
<td>X X X X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DV</td>
<td>X X X X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DA</td>
<td>X X X X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INST/PR[1:0]</td>
<td>X X X X</td>
</tr>
<tr>
<td>0x2A4 to 0x2BC</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>Reset value</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------</td>
<td>-------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>0x2C0</td>
<td>ETM_SSPCICR0</td>
<td>PC[3:0]</td>
<td>X X X X</td>
</tr>
<tr>
<td>0x2C4 to 0x30C</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x310</td>
<td>ETM_PDCR</td>
<td>PU</td>
<td>0</td>
</tr>
<tr>
<td>0x314</td>
<td>ETM_PDSR</td>
<td></td>
<td>STICKY POWER</td>
</tr>
<tr>
<td>0x318 to 0xFD8</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFA0</td>
<td>ETM_CLAIMSETR</td>
<td>CLAIMSET[3:0]</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>0xFA4</td>
<td>ETM_CLAIMCLR</td>
<td>CLAIMCLR[3:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0xFA8 to 0xFB4</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFB8</td>
<td>ETM_AUTHSTATR</td>
<td></td>
<td>X X X X X X</td>
</tr>
<tr>
<td>0xFBC</td>
<td>ETM_DEVARCHR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFC0 to 0xFD8</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFC0</td>
<td>ETM_DEVTYPEPER</td>
<td>SUBTYPE[3:0]</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>0xFD0</td>
<td>ETM_PIDR4</td>
<td>SIZE[3:0]</td>
<td>JEP106CON[3:0]</td>
</tr>
<tr>
<td>0xFD4 to 0xFD8</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFE0</td>
<td>ETM_PIDR0</td>
<td>PARTNUM[7:0]</td>
<td>0 0 1 0 0 0 0 1</td>
</tr>
<tr>
<td>0xFE4</td>
<td>ETM_PIDR1</td>
<td>JEP106ID[2:0]</td>
<td>1 0 1 1 1 0 1 1</td>
</tr>
<tr>
<td>0xFE8</td>
<td>ETM_PIDR2</td>
<td>REVISION[3:0]</td>
<td>0 0 0 1 1 0 0 1</td>
</tr>
<tr>
<td>0xFE8</td>
<td>ETM_PIDR3</td>
<td>REVAND[3:0]</td>
<td>CMOD[3:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
</tr>
</tbody>
</table>
Refer to *Table 702: Processor ROM table* for register boundary addresses.

### 58.10 Trace port interface unit (TPIU)

The TPIU formats the trace stream and outputs it on the external trace port signals. As shown in the figure below, the TPIU has two ATB slave ports for incoming trace data from the ETM and ITM respectively. The trace port is a synchronous parallel port, comprising a clock output, TRACECLK, and four data outputs, TRACEDATA(3:0). The trace port width is programmable in the range 1 to 4. Using a smaller port width reduces the number of test points/connector pins needed, and frees up IOs for other purposes, at the expenses of bandwidth restriction of the trace port, and hence of the quantity of trace information that can be output in real time.

![Figure 833. Trace port interface unit (TPIU)](image)
Trace data can also be output on the serial-wire output, TRACESWO.

For more information on the trace port interface in the Cortex®-M33, refer to the Arm® Cortex®-M33 Technical Reference Manual [5].

### 58.10.1 TPIU registers

**TPIU supported port size register (TPIU_SPSR)**

Address offset: 0x000

Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 PORTSIZE[31:0]: trace port sizes, from 1 to 32 pins

- Bit n-1 when set, indicates that port size n is supported.
- 0x0000 000F: port sizes 1 to 4 supported

**TPIU current port size register (TPIU_CPSR)**

Address offset: 0x004

Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 PORTSIZE[31:0]: current trace port size

- Bit n-1 when set, indicates that the current port size is n pins. The value of n must be within the range of supported port sizes (1-4). Only one bit can be set, or unpredictable behavior may result.
- This register must only be modified when the formatter is stopped.
**TPIU asynchronous clock prescaler register (TPIU_ACPR)**

Address offset: 0x010  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:13  Reserved, must be kept at reset value.

Bits 12:0  **PRESCALER[12:0]**  baud rate for the asynchronous output, TRACESWO  
The baud rate is given by the TRACELKIN frequency divided by (PRESCALER +1).

**TPIU selected pin protocol register (TPIU_SPPR)**

Address offset: 0x0F0  
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:2  Reserved, must be kept at reset value.

Bits 1:0  **TXMODE[1:0]**  protocol used for trace output  
0x0: parallel trace port mode  
0x1: asynchronous SWO using Manchester encoding  
0x2: asynchronous SWO using NRZ encoding  
0x3: reserved
TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300
Reset value: 0x0000 0008

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:4 Reserved, must be kept at reset value.

- **Bit 3** **FTNONSTOP**: formatter stop
  Indicates whether formatter can be stopped or not.
  1: The formatter cannot be stopped.

- **Bit 2** **TCPRESENT**: TRACECTL output pin availability
  Indicates whether the optional TRACECTL output pin is available for use.
  0: TRACECTL pin is not present in this device.

- **Bit 1** **FTSTOPPED**: formatter stop
  The formatter has received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.
  0: The formatter has not stopped.

- **Bit 0** **FLINPROG**: flush in progress
  Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.
  0: no flush in progress
  1: flush in progress

TPIU formatter and flush control register (TPIU_FFCR)

Address offset: 0x304
Reset value: 0x0000 0102

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

- **Bit 31** **FONMAN**: flush on MAN
- **Bit 30** **TRIGIN**: trace input
- **Bit 29** **ENFCONT**: enable context

3058/3119 RM0481 Rev 1
Bits 31:9  Reserved, must be kept at reset value.

Bit 8  **TRIGIN**: trigger on trigger in
  1: Indicates a trigger in the trace stream when the TRIGIN input is asserted.

Bit 7  Reserved, must be kept at reset value.

Bit 6  **FONMAN**: flush on manual
  0: flush completed
  1: Generates a flush.

Bits 5:2  Reserved, must be kept at reset value.

Bit 1  **ENFCONT**: continuous formatting enable
  Setting this bit to zero in SWO mode bypasses the formatter and only ITM/DWT trace is output. ETM trace is discarded.
  0: continuous formatting disabled
  1: continuous formatting enabled

Bit 0  Reserved, must be kept at reset value.

**TPIU periodic synchronization counter register (TPIU_PSCR)**

Address offset: 0x308

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
</tbody>
</table>

Bits 31:13  Reserved, must be kept at reset value.

Bits 12:0  **PSCOUNT[12:0]**: formatter frames counter
  Enables effective use of different sized TPAs without wasting large amounts of the storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word synchronization frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.
TPIU claim tag set register (TPIU_CLAIMSETR)

Address offset: 0xFA0
Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 CLAIMSET[3:0]: claim tag bits setting
Write:
0000: no effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 CLAIMCLR[3:0]: claim tag bits reset
Write:
0000: no effect
xxx1: Clears bit 0.
xx1x: Clears bit 1.
x1xx: Clears bit 2.
1xxx: Clears bit 3.
Read: Returns current value of claim tag.
TPIU device configuration register (TPIU_DEVIDR)

Address offset: 0xFC8  
Reset value: 0x0000 0CA1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:12 Reserved, must be kept at reset value.

- Bit 11 **SWOUARTNRZ**: Serial-wire output, NRZ support  
  0x1: supported

- Bit 10 **SWOMAN**: Serial-wire output, Manchester encoded format, support  
  0x1: supported

- Bit 9 **TCLKDATA**: trace clock plus data support  
  0x0: supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2  
  0x2: FIFO size = 4 bytes

- Bit 5 **CLKRELAT**: ATB clock and TRACECLKIN relationship (synchronous or asynchronous)  
  0x1: asynchronous

Bits 4:0 **MAXNUM[4:0]**: number/type of ATB input port multiplexing  
  0x1: two input ports

TPIU device type identifier register (TPIU_DEVTYPER)

Address offset: 0xFCC  
Reset value: 0x0000 0011

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **SUBTYPE[3:0]**
- **MAJORTYPE[3:0]**

- **r**: read-only
- **w**: write-only
- **r**: read-write

TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 15:12</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **SUBTYPE[3:0]**: sub-classification
0x1: trace port component

Bits 3:0  **MAJORTYPE[3:0]**: major classification
0x1: trace sink component

TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 0021

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 15:12</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0  **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 0021

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 15:8</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PARTNUM[7:0]**: part number bits [7:0]
0x21: TPIU part number
**TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)**

Address offset: 0xFE4

Reset value: 0x0000 00BD

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
   0xB: Arm® JEDEC code

Bits 3:0  **PARTNUM[11:8]**: part number bits [11:8]
   0xD: TPIU part number

**TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)**

Address offset: 0xFE8

Reset value: 0x0000 000B

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **REVISION[3:0]**: component revision number
   0x0: r0p0

Bit 3  **JEDEC**: JEDEC assigned value
   0x1: designer identification specified by JEDEC

   0x3: Arm® JEDEC code
TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 REVAND[3:0]: metal fix version
0x0: no metal fix

Bits 3:0 CMOD[3:0]: customer modified
0x0: no customer modifications

TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 PREAMBLE[7:0]: component identification bits [7:0]
0x0D: common identification value
### TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4  
Reset value: 0x0000 0090

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**Bits 31:8** Reserved, must be kept at reset value.

0x09: CoreSight™ component

0x00: common identification value

### TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8  
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:8** Reserved, must be kept at reset value.

**Bits 7:0** `PREAMBLE[19:12]`: component identification bits [23:16]  
0x05: common identification value
### TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:8: Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** component identification bits [31:24]

0xB1: common identification value

### 58.10.2 TPIU register map

#### Table 710. TPIU register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>SWOARTNRZ</th>
<th>SLOWMAN</th>
<th>TCLKDATA</th>
<th>FIFO/SIZE[2:0]</th>
<th>CURRENT</th>
<th>MAXNUM[4:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x30C to 0xF9C</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xAF0</td>
<td>TPIU_CLAIMSETR</td>
<td></td>
<td></td>
<td></td>
<td>CLAIMSET [3:0]</td>
<td>1 1 1 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFA4</td>
<td>TPIU_CLAIMCLR</td>
<td></td>
<td></td>
<td></td>
<td>CLAIMCLR [3:0]</td>
<td>0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFA8 to 0xFC4</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFC8</td>
<td>TPIU_DEVIDR</td>
<td></td>
<td></td>
<td></td>
<td>SUBTYPE [3:0]</td>
<td>1 1 1 0 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td>MAJOR TYPE [3:0]</td>
<td>0 0 0 0 1</td>
<td></td>
</tr>
<tr>
<td>0xFC0</td>
<td>TPIU_DEVTYPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD0</td>
<td>TPIU_PIDR4</td>
<td></td>
<td></td>
<td></td>
<td>SIZE [3:0]</td>
<td>0 0 0 1 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td>JEP106CON [3:0]</td>
<td>1 0 1 0</td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>TPIU_PIDR0</td>
<td></td>
<td></td>
<td></td>
<td>PARTNUM[7:0]</td>
<td>0 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE4</td>
<td>TPIU_PIDR1</td>
<td></td>
<td></td>
<td></td>
<td>JEP106ID [3:0]</td>
<td>0 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td>PARTNUM [11:8]</td>
<td>1 1 0 1 0</td>
<td></td>
</tr>
<tr>
<td>0xFE8</td>
<td>TPIU_PIDR2</td>
<td></td>
<td></td>
<td></td>
<td>REVISION [3:0]</td>
<td>0 1 0 1 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td>JEP106ID [6:4]</td>
<td>0 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td>0xFEC</td>
<td>TPIU_PIDR3</td>
<td></td>
<td></td>
<td></td>
<td>REVAND[3:0]</td>
<td>0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td>CMOD[3:0]</td>
<td>0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>0xFF0</td>
<td>TPIU_CIDR0</td>
<td></td>
<td></td>
<td></td>
<td>PREAMBLE[7:0]</td>
<td>0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 1 0 1 0</td>
<td></td>
</tr>
<tr>
<td>0xFF4</td>
<td>TPIU_CIDR1</td>
<td></td>
<td></td>
<td></td>
<td>CLASS[3:0]</td>
<td>0 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td>PREAMBLE [11:8]</td>
<td>1 0 0 1 0</td>
<td></td>
</tr>
<tr>
<td>0xFF8</td>
<td>TPIU_CIDR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td>PREAMBLE[19:12]</td>
<td>1 0 0 1 0</td>
<td></td>
</tr>
<tr>
<td>0xFFC</td>
<td>TPIU_CIDR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 0 0 1 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td>PREAMBLE[27:20]</td>
<td>1 0 0 1 0</td>
<td></td>
</tr>
</tbody>
</table>

Refer to *Table 701: MCU ROM table* for register boundary addresses.
58.11 Cross-trigger interface (CTI)

The CTI allows cross triggering between the processor and the ETM (see the figure below).

The CTI enables events from various sources to trigger debug and/or trace activity. For example, a watchpoint reached in the processor can start or stop code trace, or a trace comparator can halt the processor.

The trigger input and output signals for the CTI are listed in the tables below.

**Table 711. CTI inputs**

<table>
<thead>
<tr>
<th>Number</th>
<th>Source signal</th>
<th>Source component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HALTED</td>
<td>CPU</td>
<td>Processor halted - CPU is in debug mode</td>
</tr>
<tr>
<td>1</td>
<td>ETMTRIGGER0</td>
<td>DWT</td>
<td>DWT comparator output 0</td>
</tr>
<tr>
<td>2</td>
<td>ETMTRIGGER1</td>
<td>DWT</td>
<td>DWT comparator output 1</td>
</tr>
<tr>
<td>3</td>
<td>ETMTRIGGER2</td>
<td>DWT</td>
<td>DWT comparator output 2</td>
</tr>
<tr>
<td>4</td>
<td>ETMTRIGOUT0</td>
<td>ETM</td>
<td>ETM event output 0</td>
</tr>
<tr>
<td>5</td>
<td>ETMTRIGOUT1</td>
<td>ETM</td>
<td>ETM event output 1</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
</tbody>
</table>

**Table 712. CTI outputs**

<table>
<thead>
<tr>
<th>Number</th>
<th>Source signal</th>
<th>Destination component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EDBGRQ</td>
<td>CPU</td>
<td>CPU halt request - Puts CPU in debug mode</td>
</tr>
<tr>
<td>1</td>
<td>DBGRESTART</td>
<td>CPU</td>
<td>CPU restart request - CPU exits debug mode</td>
</tr>
</tbody>
</table>
For more information on the cross-trigger interface CoreSight™ component, refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual [2].

### 58.11.1 CTI registers

The register file base address for the CTI is 0xE004 2000.

#### CTI control register (CTI_CONTROLR)

Address offset: 0x000  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:1</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>
| 0 | **GLBEN**: global CTI enable | 0: disabled  
1: enabled |

#### CTI trigger acknowledge register (CTI_INTACKR)

Address offset: 0x010  
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:1</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td><strong>INTACK[7:0]</strong></td>
<td>w w w w w w w w</td>
</tr>
</tbody>
</table>
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **INTACK[7:0]**: trigger acknowledge  
There is one bit of the register for each CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.

**CTI application trigger set register (CTI_APPSETR)**  
Address offset: 0x014  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**APPSET[3:0]**: channel event setting  
Read:  
XXX0: channel 0 event inactive  
XXX0: channel 0 event active  
XX0X: channel 1 event inactive  
XX1X: channel 1 event active  
X0XX: channel 2 event inactive  
X1XX: channel 2 event active  
0XXX: channel 3 event inactive  
1XXX: channel 3 event active  
Write:  
XXX0: no effect  
XXX0: Sets event on channel 0.  
XX0X: no effect  
XX1X: Sets event on channel 1.  
X0XX: no effect  
X1XX: Sets event on channel 2.  
0XXX: no effect  
1XXX: Sets event on channel 3.
### CTI application trigger clear register (CTI_APPCLEAR)

Address offset: 0x018  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**APPCLEAR[3:0]**  
- Bits 31:4 Reserved, must be kept at reset value.  
- Bits 3:0 **APPCLEAR[3:0]**: channel event clear  
  - 0000: no effect  
  - XXX1: Clears event on channel 0.  
  - XX1X: Clears event on channel 1.  
  - X1XX: Clears event on channel 2.  
  - 1XXX: Clears event on channel 3.

### CTI application pulse register (CTI_APPPULSER)

Address offset: 0x01C  
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**APPULSE[3:0]**  
- Bits 31:4 Reserved, must be kept at reset value.  
- Bits 3:0 **APPULSE[3:0]**: pulse channel event  
  - This register clears itself immediately.  
  - 0000: no effect  
  - XXX1: Generates pulse on channel 0.  
  - XX1X: Generates pulse on channel 1.  
  - X1XX: Generates pulse on channel 2.  
  - 1XXX: Generates pulse on channel 3.
**CTI trigger input x enable register (CTI_INENxR)**

Address offset: 0x020 + 0x004 * x, (x = 0 to 7)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **TRIGINEN[3:0]**: trigger input event enable

Enables or disables a cross trigger event on each of the four channels when CTITRIGINx is activated (x = 0 to 7).

0000: Trigger does not generate events on channels.
XXX1: Trigger x generates events on channel 0.
XX1X: Trigger x generates events on channel 1.
X1XX: Trigger x generates events on channel 2.
1XXX: Trigger x generates events on channel 3.

**CTI trigger output x enable register (CTI_OUTENxR)**

Address offset: 0x0A0 + 0x004 * x, (x = 0 to 7)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **TRIGOUTEN[3:0]**: trigger output event enable

For each channel, defines whether an event on that channel generates a trigger on CTITRIGOUTx (x = 0 to 7).

0000: Channel events do not generate triggers on trigger outputs.
XXX1: Channel 0 events generate triggers on trigger output x.
XX1X: Channel 1 events generate triggers on trigger output x.
X1XX: Channel 2 events generate triggers on trigger output x.
1XXX: Channel 3 events generate triggers on trigger output x.
### CTI trigger input status register (CTI_TRGISTSR)

Address offset: 0x130  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>TRIGINSTATUS[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8: Reserved, must be kept at reset value.

Bits 7:0 **TRIGINSTATUS[7:0]:** trigger input status  
There is one bit of the register for each CTITRIGINx input. When a bit is set to 1, it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

### CTI trigger output status register (CTI_TRGOSTSR)

Address offset: 0x134  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>TRIGOUTSTATUS[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8: Reserved, must be kept at reset value.

Bits 7:0 **TRIGOUTSTATUS[7:0]:** trigger output status  
There is one bit of the register for each CTITRIGOUT output. When a bit is set to 1, it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.
CTI channel input status register (CTI_CHINSTSR)

Address offset: 0x138
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **CHINSTATUS[3:0]**: channel input status

There is one bit of the register for each channel input. When a bit is set to 1 it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

CTI channel output status register (CTI_CHOUTSTSR)

Address offset: 0x13C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **CHOUTSTATUS[3:0]**: channel output status

There is one bit of the register for each channel output. When a bit is set to 1 it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.
CTI channel gate register (CTI_GATER)

Address offset: 0x140
Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 GATEEN[3:0]: channel output enable
For each channel, defines whether an event on that channel can propagate over the CTM to other CTIs.
0000: Channels events do not propagate.
XXX1: Channel 0 events propagate.
XX1X: Channel 1 events propagate.
X1XX: Channel 2 events propagate.
1XXX: Channel 3 events propagate.

CTI device configuration register (CTI_DEVIDR)

Address offset: 0xFC8
Reset value: 0x0004 0800

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 NUMCH[3:0]: number of ECT channels available
0x4: four channels

Bits 15:8 NUMTRIG[7:0]: number of ECT triggers available
0x8: height trigger inputs and height trigger outputs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 EXTMUXNUM[4:0]: number of trigger input/output multiplexers
0x0: none
CTI device type identifier register (CTI_DEVTYPEPER)

Address offset: 0xFCC
Reset value: 0x0000 0014

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **SUBTYPE[3:0]**: sub-classification
0x1: cross-triggering component.

Bits 3:0  **MAJORTYPE[3:0]**: major classification
0x4: Indicates that this component allows a debugger to control other components in a CoreSight™ SoC-400 system.

CTI CoreSight peripheral identity register 4 (CTI_PIDR4)

Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0  **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code
### CTI CoreSight peripheral identity register 0 (CTI_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 0021

| Bits 31:8 | Reserved, must be kept at reset value. |
| Bits 7:0  | **PARTNUM[7:0]**: part number bits [7:0] |
|          | 0x21: CTI part number |

| Bits 31:8 | Reserved, must be kept at reset value. |
| Bits 7:4  | **JEP106ID[3:0]**: JEP106 identity code bits [3:0] |
|          | 0xB: Arm® JEDEC code |
| Bits 3:0  | **PARTNUM[11:8]**: part number bits [11:8] |
|          | 0xD: CTI part number |
**CTI CoreSight peripheral identity register 2 (CTI_PIDR2)**

Address offset: 0xFE8

Reset value: 0x0000 000B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
- 0x0: r0p0
- Bit 3 **JEDEC**: JEDEC assigned value
  - 0x1: designer identification specified by JEDEC

- 0x3: Arm® JEDEC code

**CTI CoreSight peripheral identity register 3 (CTI_PIDR3)**

Address offset: 0xFEC

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
- 0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
- 0x0: no customer modifications
CTI CoreSight component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0  **PREAMBLE[7:0]**: component identification bits [7:0]
  0x0D: common identification value

CTI CoreSight peripheral identity register 1 (CTI_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 0090

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:4  **CLASS[3:0]**: component identification bits [15:12] - component class
  0x09: CoreSight™ component
Bits 3:0  **PREAMBLE[11:8]**: component identification bits [11:8]
  0x0: common identification value
CTI CoreSight component identity register 2 (CTI_CIDR2)
Address offset: 0xFF8
Reset value: 0x0000 0005

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

CTI CoreSight component identity register 3 (CTI_CIDR3)
Address offset: 0xFFC
Reset value: 0x0000 00B1

Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

58.11.2  CTI register map

Table 713. CTI register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
<th>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>r r r r r r r r</td>
</tr>
<tr>
<td>0x004</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>r r r r r r r r</td>
</tr>
</tbody>
</table>

INTACK[7:0]
Table 713. CTI register map and reset values (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x014</td>
<td>CTI_APPSETR</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x018</td>
<td>CTI_APPCLEAR</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x01C</td>
<td>CTI_APPPULSER</td>
<td>X X X X</td>
<td></td>
<td></td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x020 to 0x03C</td>
<td>CTI_INEN0R to CTI_INEN7R</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x040 to 0x09C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0A0 to 0x0BC</td>
<td>CTI_OUTEN0R to CTI_OUTEN7R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C0 to 0x12C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x130</td>
<td>CTI_TRGINSTSR</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x134</td>
<td>CTI_TRGOSTSR</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x138</td>
<td>CTI_CHININSTSR</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x13C</td>
<td>CTI_CHOUTSTSR</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x140</td>
<td>CTI_GATER</td>
<td>1 1 1 1</td>
<td></td>
<td></td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x144 to 0x1FC4</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFC8</td>
<td>CTI_DEVIDR</td>
<td>0 1 0 0</td>
<td>0 0 0 1</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0xFC0</td>
<td>CTI_DEVTYPE</td>
<td>0 0 0 1</td>
<td>0 1 0 0</td>
<td>0 1 0 0</td>
<td>0 1 0 0</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>0xFD0</td>
<td>CTI_PIDR4</td>
<td>0 0 0 0</td>
<td>0 1 0 0</td>
<td>0 1 0 0</td>
<td>0 1 0 0</td>
<td>0 1 0 0</td>
</tr>
</tbody>
</table>
Refer to Table 702: Processor ROM table for register boundary addresses.

### 58.12 Microcontroller debug unit (DBGMCU)

The DBGMCU is a component containing a number of registers that control the power and clock behavior in debug mode. It allows the debugger (or the software) to:

- maintain the clock and power to the processor cores when in low-power modes (Sleep, Stop or Standby)
- maintain the clock and power to the system debug and trace components when in low-power modes
- stop the clock to certain peripherals (SMBUS timeout, watchdogs, timers, RTC) when either processor core is stopped in debug mode

#### 58.12.1 Device ID

The DBGMCU includes an identity code register, DBGMCU_IDCODE. This register contains the ID code for the device. Debug tools can locate this register via the CoreSight™ discovery procedure described in Section 58.5: ROM tables.

#### 58.12.2 Low-power mode emulation

When the device enters either Stop mode (clocks are stopped) or Standby mode (core power is switched off), the debugger can no longer access the debug access port and loses the connection with the device. To avoid this, the debugger (or software) can set the
DBG_STANDBY and/or DBG_STOP bits in the **DBGMCU configuration register (DBGMCU_CR)**. These bits, when set, maintain the clock and power to the processor while the device is in the corresponding low-power mode. The processor remains in Sleep mode, and exits the low-power mode in the normal way. However, peripheral devices continue to operate, so the device behaviour may not be identical to that of the actual low-power mode.

### 58.12.3 Peripheral clock freeze

The DBGMCU peripheral clock freeze registers allow the operation of certain peripherals to be suspended in debug mode. The peripheral units which support this feature are listed in the table below.

<table>
<thead>
<tr>
<th>Bus</th>
<th>Control register</th>
<th>Peripheral</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APB1L</td>
<td>DBGMCU_APB1LFZR</td>
<td>I3C1</td>
<td>I3C1 SCL stall timeout counter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I2C2</td>
<td>I2C2 SMBUS timeout</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I2C1</td>
<td>I2C1 SMBUS timeout</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IWDG</td>
<td>Independent watchdog</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WWDG</td>
<td>Window watchdog</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM14</td>
<td>General purpose timer 14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM13</td>
<td>General purpose timer 13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM12</td>
<td>General purpose timer 12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM7</td>
<td>General purpose timer 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM6</td>
<td>General purpose timer 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM5</td>
<td>General purpose timer 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM4</td>
<td>General purpose timer 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM3</td>
<td>General purpose timer 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM2</td>
<td>General purpose timer 2</td>
</tr>
<tr>
<td>APB1H</td>
<td>DGBMCU_APB1HFZR</td>
<td>LPTIM2</td>
<td>Low power timer 2</td>
</tr>
<tr>
<td>APB2</td>
<td>DBGMCU_APB2FZR</td>
<td>TIM17</td>
<td>General purpose timer 17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM16</td>
<td>General purpose timer 16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM15</td>
<td>General purpose timer 15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM8</td>
<td>General purpose timer 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM1</td>
<td>General purpose timer 1</td>
</tr>
</tbody>
</table>
Each peripheral unit or DMA channel has a corresponding control bit, DBG_xxx_STOP, where xxx is the acronym of the peripheral (or DMA channel). The control bits are organized in DBGMCU_zzzFZR registers, where zzz corresponds to the name of the bus (AHB or APB). For example, DBGMCU_APB1LFZR contains the control bits for peripherals on the APB1L bus.

The control bit, when set, causes the corresponding peripheral operation to be suspended when the CPU is stopped in debug (HALTED = 1), according to the table below:

<table>
<thead>
<tr>
<th>HALTED</th>
<th>DBG_xxx_STOP</th>
<th>Peripheral behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>The operation continues.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>The operation continues.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>The operation is suspended.</td>
</tr>
</tbody>
</table>

The accessibility of the bits DBG_xxx_STOP by the debugger depends on the state of the authentication signal spiden.

When spiden = 1 (secure privilege debug enabled), all bits can be modified and read by both debugger and software (secure or non-secure).

When spiden = 0 (secure privilege debug disabled), only bits corresponding to non-secure peripherals (or DMA channels) can be modified by debugger or software. All bits can be read.

The status (secure or non-secure) of a TrustZone–aware peripheral or a DMA channel, is signaled to the DBGMCU by the peripheral.

### Table 714. Peripheral clock freeze control bits (continued)

<table>
<thead>
<tr>
<th>Bus</th>
<th>Control register</th>
<th>Peripheral</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APB3</td>
<td>DBGMCU_APB3FZR</td>
<td>RTC</td>
<td>Real time clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM6</td>
<td>Low power timer 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM5</td>
<td>Low power timer 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM4</td>
<td>Low power timer 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM3</td>
<td>Low power timer 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM1</td>
<td>Low power timer 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I2C4</td>
<td>I2C4 SMBUS timeout</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I2C3</td>
<td>I2C3 SMBUS timeout</td>
</tr>
<tr>
<td>AHB1</td>
<td>DBGMCU_AHB1FZR</td>
<td>GPDMA2 0 to 7</td>
<td>General purpose DMA2 channels 0 to 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPDMA1 0 to 7</td>
<td>General purpose DMA1 channels 0 to 7</td>
</tr>
</tbody>
</table>

### Table 715. Peripheral behaviour in debug mode
58.12.4 **DBGMCU registers**

The DBGMCU registers are not reset by a system reset, only by a power-on reset. They are accessible to the debugger via the AHB access port at base address 0xE004 4000, and to software at base address 0x4402 4000.

**DBGMCU identity code register (DBGMCU_IDCODE)**

Address offset: 0x00  
Reset value: 0xXXXX 6XXX  
This register is always accessible.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-16</td>
<td>REV_ID[15:0]</td>
<td>0xXXXX 6XXX</td>
</tr>
<tr>
<td>11-0</td>
<td>DEV_ID[11:0]</td>
<td></td>
</tr>
<tr>
<td>Bits 31:16</td>
<td>REV_ID[15:0]</td>
<td>Revision</td>
</tr>
<tr>
<td>A:</td>
<td>0x1000</td>
<td></td>
</tr>
<tr>
<td>Z:</td>
<td>0x1001</td>
<td></td>
</tr>
<tr>
<td>X:</td>
<td>0x1007</td>
<td></td>
</tr>
<tr>
<td>Bits 15:12</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 11:0</td>
<td>DEV_ID[11:0]</td>
<td>Device identification</td>
</tr>
<tr>
<td>0x484</td>
<td>STM32H563/H573 and STM32H562</td>
<td></td>
</tr>
</tbody>
</table>

**DBGMCU configuration register (DBGMCU_CR)**

Address offset: 0x04  
Reset value: 0x0000 0000  
This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, debugger accesses are ignored.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-16</td>
<td>DCRT</td>
<td>rw</td>
</tr>
<tr>
<td>20-16</td>
<td>TRACE_MODE[3:0]</td>
<td></td>
</tr>
<tr>
<td>15-11</td>
<td>TRACE_EN</td>
<td>rw</td>
</tr>
<tr>
<td>10-6</td>
<td>TRACE_IOEN</td>
<td>rw</td>
</tr>
<tr>
<td>5-1</td>
<td>DBG_STANDBY</td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>DBG_STOP</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:17  Reserved, must be kept at reset value.

Bit 16  **DCRT**: Debug credentials reset type
This bit selects which type of reset is used to revoke the debug authentication credentials
- 0: System reset
- 1: Power reset

Bits 15:8  Reserved, must be kept at reset value.

Bits 7:6  **TRACE_MODE[1:0]**: trace pin assignment
- 0x0: trace pins assigned for asynchronous mode (TRACESWO)
- 0x1: trace pins assigned for synchronous mode with a port width of 1 (TRACECK, TRACED0)
- 0x2: trace pins assigned for synchronous mode with a port width of 2 ((TRACECK, TRACED0-1)
- 0x3: trace pins assigned for synchronous mode with a port width of 4 ((TRACECK, TRACED0-3)

Bit 5  **TRACE_EN**: trace port and clock enable.
This bit enables the trace port clock, TRACECK.
- 0: disabled
- 1: enabled

Bit 4  **TRACE_IOEN**: trace pin enable
- 0: disabled - trace pins not assigned
- 1: enabled - trace pins assigned according to the value of TRACE_MODE field

Bit 3  Reserved, must be kept at reset value.

Bit 2  **DBG_STANDBY**: Allows debug in Standby mode
- 0: normal operation
  - All clocks are disabled and the core powered down automatically in Standby mode.
  - 1: automatic clock stop/power down disabled
  - All active clocks and oscillators continue to run during Standby mode, and the core supply is maintained, allowing full debug capability. On exit from Standby mode, a system reset is performed.

Bit 1  **DBG_STOP**: Allows debug in Stop mode
- 0: normal operation
  - All clocks are disabled automatically in Stop mode.
- 1: automatic clock stop disabled
  - All active clocks and oscillators continue to run during Stop mode, allowing full debug capability. On exit from Stop mode, the clock settings are set to the Stop mode exit state.

Bit 0  Reserved, must be kept at reset value.

**DBGMCU APB1L peripheral freeze register (DBGMCU_APB1LFZR)**

Address offset: 0x08
Reset value: 0x0000 0000
This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, accesses are ignored.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBG_I3C1_STOP</td>
<td>DBG_I2C2_STOP</td>
<td>DBG_I2C1_STOP</td>
<td>DBG_IWDG_STOP</td>
<td>DBG_WWDG_STOP</td>
<td>DBG_TIM14_STOP</td>
<td>DBG_TIM13_STOP</td>
<td>DBG_TIM12_STOP</td>
<td>DBG_TIM7_STOP</td>
<td>DBG_TIM6_STOP</td>
<td>DBG_TIM5_STOP</td>
<td>DBG_TIM4_STOP</td>
<td>DBG_TIM3_STOP</td>
<td>DBG_TIM2_STOP</td>
<td>DBG_TIM14_STOP</td>
<td>DBG_TIM13_STOP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

- **Bit 23 DBG_I3C1_STOP**: I3C1 SCL stall counter stop in debug
  - 0: normal operation. I3C1 SCL stall timeout counter continues to operate while CPU is in debug mode.
  - 1: stop in debug. I3C1 SCL stall timeout counter is frozen while CPU is in debug mode.

- **Bit 22 DBG_I2C2_STOP**: I2C2 SMBUS timeout stop in debug
  - 0: normal operation. I2C2 SMBUS timeout continues to operate while CPU is in debug mode.
  - 1: stop in debug. I2C2 SMBUS timeout is frozen while CPU is in debug mode.

- **Bit 21 DBG_I2C1_STOP**: I2C1 SMBUS timeout stop in debug
  - 0: normal operation. I2C1 SMBUS timeout continues to operate while CPU is in debug mode.
  - 1: stop in debug. I2C1 SMBUS timeout is frozen while CPU is in debug mode.

Bits 20:13 Reserved, must be kept at reset value.

- **Bit 12 DBG_IWDG_STOP**: IWDG stop in debug
  - 0: normal operation. IWDG continues to operate while CPU is in debug mode.
  - 1: stop in debug. IWDG is frozen while CPU is in debug mode.

- **Bit 11 DBG_WWDG_STOP**: WWDG stop in debug
  - 0: normal operation. WWDG continues to operate while CPU is in debug mode.
  - 1: stop in debug. WWDG is frozen while CPU is in debug mode.

Bits 10:9 Reserved, must be kept at reset value.

- **Bit 8 DBG_TIM14_STOP**: TIM14 stop in debug
  - 0: normal operation. TIM14 continues to operate while CPU is in debug mode.
  - 1: stop in debug. TIM14 is frozen while CPU is in debug mode.

- **Bit 7 DBG_TIM13_STOP**: TIM13 stop in debug
  - 0: normal operation. TIM13 continues to operate while CPU is in debug mode.
  - 1: stop in debug. TIM13 is frozen while CPU is in debug mode.

- **Bit 6 DBG_TIM12_STOP**: TIM12 stop in debug
  - 0: normal operation. TIM12 continues to operate while CPU is in debug mode.
  - 1: stop in debug. TIM12 is frozen while CPU is in debug mode.
### DBGMCU APB1H peripheral freeze register (DBGMCU_APB1HFZR)

Address offset: 0x0C  
Reset value: 0x0000 0000  
This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, accesses are ignored.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**Bit 5** DBG_TIM7_STOP: TIM7 stop in debug  
0: normal operation. TIM7 continues to operate while CPU is in debug mode.  
1: stop in debug. TIM7 is frozen while CPU is in debug mode.

**Bit 4** DBG_TIM6_STOP: TIM6 stop in debug  
0: normal operation. TIM6 continues to operate while CPU is in debug mode.  
1: stop in debug. TIM6 is frozen while CPU is in debug mode.

**Bit 3** DBG_TIM5_STOP: TIM5 stop in debug  
0: normal operation. TIM5 continues to operate while CPU is in debug mode.  
1: stop in debug. TIM5 is frozen while CPU is in debug mode.

**Bit 2** DBG_TIM4_STOP: TIM4 stop in debug  
0: normal operation. TIM4 continues to operate while CPU is in debug mode.  
1: stop in debug. TIM4 is frozen while CPU is in debug mode.

**Bit 1** DBG_TIM3_STOP: TIM3 stop in debug  
0: normal operation. TIM3 continues to operate while CPU is in debug mode.  
1: stop in debug. TIM3 is frozen while CPU is in debug mode.

**Bit 0** DBG_TIM2_STOP: TIM2 stop in debug  
0: normal operation. TIM2 continues to operate while CPU is in debug mode.  
1: stop in debug. TIM2 is frozen while CPU is in debug mode.

Bits 31:6 Reserved, must be kept at reset value.

**Bit 5** DBG_LPTIM2_STOP: LPTIM2 stop in debug  
0: normal operation. LPTIM2 continues to operate while CPU is in debug mode.  
1: stop in debug. LPTIM2 is frozen while CPU is in debug mode.

Bits 4:0 Reserved, must be kept at reset value.
DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZR)

Address offset: 0x10
Reset value: 0x0000 0000

This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, accesses are ignored.

| Bit 31:19 Reserved, must be kept at reset value. |
| Bit 18 DBG_TIM17_STOP: TIM17 stop in debug |
| 0: normal operation. TIM17 continues to operate while CPU is in debug mode. |
| 1: stop in debug. TIM17 is frozen while CPU is in debug mode. |
| Bit 17 DBG_TIM16_STOP: TIM16 stop in debug |
| 0: normal operation. TIM16 continues to operate while CPU is in debug mode. |
| 1: stop in debug. TIM16 is frozen while CPU is in debug mode. |
| Bit 16 DBG_TIM15_STOP: TIM15 stop in debug |
| 0: normal operation. TIM15 continues to operate while CPU is in debug mode. |
| 1: stop in debug. TIM15 is frozen while CPU is in debug mode. |

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 DBG_TIM8_STOP: TIM8 stop in debug
0: normal operation. TIM8 continues to operate while CPU is in debug mode.
1: stop in debug. TIM8 is frozen while CPU is in debug mode.

Bit 12 Reserved, must be kept at reset value.

Bit 11 DBG_TIM1_STOP: TIM1 stop in debug
0: normal operation. TIM1 continues to operate while CPU is in debug mode.
1: stop in debug. TIM1 is frozen while CPU is in debug mode.

Bits 10:0 Reserved, must be kept at reset value.

DBGMCU APB3 peripheral freeze register (DBGMCU_APB3FZR)

Address offset: 0x14
Reset value: 0x0000 0000
This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, accesses are ignored.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>DBG_RTC_STOP: RTC stop in debug</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>DBG_RTC_STOP: RTC stop in debug</td>
<td>0: normal operation. RTC continues to operate while CPU is in debug mode. 1: stop in debug. RTC is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>29</td>
<td>DBG_LPTIM6_STOP: LPTIM6 stop in debug</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>DBG_LPTIM6_STOP: LPTIM6 stop in debug</td>
<td>0: normal operation. LPTIM6 continues to operate while CPU is in debug mode. 1: stop in debug. LPTIM6 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>27</td>
<td>DBG_LPTIM5_STOP: LPTIM5 stop in debug</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>DBG_LPTIM5_STOP: LPTIM5 stop in debug</td>
<td>0: normal operation. LPTIM5 continues to operate while CPU is in debug mode. 1: stop in debug. LPTIM5 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>25</td>
<td>DBG_LPTIM4_STOP: LPTIM4 stop in debug</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>DBG_LPTIM4_STOP: LPTIM4 stop in debug</td>
<td>0: normal operation. LPTIM4 continues to operate while CPU is in debug mode. 1: stop in debug. LPTIM4 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>23</td>
<td>DBG_LPTIM3_STOP: LPTIM3 stop in debug</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>DBG_LPTIM3_STOP: LPTIM3 stop in debug</td>
<td>0: normal operation. LPTIM3 continues to operate while CPU is in debug mode. 1: stop in debug. LPTIM3 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>21</td>
<td>DBG_LPTIM1_STOP: LPTIM1 stop in debug</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>DBG_LPTIM1_STOP: LPTIM1 stop in debug</td>
<td>0: normal operation. LPTIM1 continues to operate while CPU is in debug mode. 1: stop in debug. LPTIM1 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>19</td>
<td>DBG_LPTIM1_STOP: LPTIM1 stop in debug</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>DBG_LPTIM1_STOP: LPTIM1 stop in debug</td>
<td>0: normal operation. LPTIM1 continues to operate while CPU is in debug mode. 1: stop in debug. LPTIM1 is frozen while CPU is in debug mode.</td>
</tr>
</tbody>
</table>

Bits 16:12 Reserved, must be kept at reset value.
**DBGMCU AHB1 peripheral freeze register (DBGMCU_AHB1FZR)**

Address offset: 0x20
Reset value: 0x0000 0000

This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, accesses are ignored.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value in Debug Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>DBG_I2C4_STOP</td>
<td>0: normal operation. I2C4 SMBUS timeout counter continues to operate while CPU is in debug mode. 1: stop in debug. I2C4 SMBUS timeout counter is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>10</td>
<td>DBG_I2C3_STOP</td>
<td>0: normal operation. I2C3 SMBUS timeout counter continues to operate while CPU is in debug mode. 1: stop in debug. I2C3 SMBUS timeout counter is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>9-0</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value in Debug Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>DBG_GPDMA2_7_STOP</td>
<td>0: normal operation. GPDMA2 channel 7 continues to operate while CPU is in debug mode. 1: stop in debug. GPDMA2 channel 7 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>22</td>
<td>DBG_GPDMA2_6_STOP</td>
<td>0: normal operation. GPDMA2 channel 6 continues to operate while CPU is in debug mode. 1: stop in debug. GPDMA2 channel 6 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>21</td>
<td>DBG_GPDMA2_5_STOP</td>
<td>0: normal operation. GPDMA2 channel 5 continues to operate while CPU is in debug mode. 1: stop in debug. GPDMA2 channel 5 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>20-0</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
</tbody>
</table>
### Debug support (DBG) RM0481

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value 0</th>
<th>Value 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td><strong>DBG_GPDMA2_4_STOP</strong>: GPDMA2 channel 4 stop in debug</td>
<td>normal operation. GPDMA2 channel 4 continues to operate while CPU is in debug mode.</td>
<td>stop in debug. GPDMA2 channel 4 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>19</td>
<td><strong>DBG_GPDMA2_3_STOP</strong>: GPDMA2 channel 3 stop in debug</td>
<td>normal operation. GPDMA2 channel 3 continues to operate while CPU is in debug mode.</td>
<td>stop in debug. GPDMA2 channel 3 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>18</td>
<td><strong>DBG_GPDMA2_2_STOP</strong>: GPDMA2 channel 2 stop in debug</td>
<td>normal operation. GPDMA2 channel 2 continues to operate while CPU is in debug mode.</td>
<td>stop in debug. GPDMA2 channel 2 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>17</td>
<td><strong>DBG_GPDMA2_1_STOP</strong>: GPDMA2 channel 1 stop in debug</td>
<td>normal operation. GPDMA2 channel 1 continues to operate while CPU is in debug mode.</td>
<td>stop in debug. GPDMA2 channel 1 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>16</td>
<td><strong>DBG_GPDMA2_0_STOP</strong>: GPDMA2 channel 0 stop in debug</td>
<td>normal operation. GPDMA2 channel 0 continues to operate while CPU is in debug mode.</td>
<td>stop in debug. GPDMA2 channel 0 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>15:8</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td><strong>DBG_GPDMA1_7_STOP</strong>: GPDMA1 channel 7 stop in debug</td>
<td>normal operation. GPDMA1 channel 7 continues to operate while CPU is in debug mode.</td>
<td>stop in debug. GPDMA1 channel 7 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>6</td>
<td><strong>DBG_GPDMA1_6_STOP</strong>: GPDMA1 channel 6 stop in debug</td>
<td>normal operation. GPDMA1 channel 6 continues to operate while CPU is in debug mode.</td>
<td>stop in debug. GPDMA1 channel 6 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>5</td>
<td><strong>DBG_GPDMA1_5_STOP</strong>: GPDMA1 channel 5 stop in debug</td>
<td>normal operation. GPDMA1 channel 5 continues to operate while CPU is in debug mode.</td>
<td>stop in debug. GPDMA1 channel 5 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>4</td>
<td><strong>DBG_GPDMA1_4_STOP</strong>: GPDMA1 channel 4 stop in debug</td>
<td>normal operation. GPDMA1 channel 4 continues to operate while CPU is in debug mode.</td>
<td>stop in debug. GPDMA1 channel 4 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>3</td>
<td><strong>DBG_GPDMA1_3_STOP</strong>: GPDMA1 channel 3 stop in debug</td>
<td>normal operation. GPDMA1 channel 3 continues to operate while CPU is in debug mode.</td>
<td>stop in debug. GPDMA1 channel 3 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>2</td>
<td><strong>DBG_GPDMA1_2_STOP</strong>: GPDMA1 channel 2 stop in debug</td>
<td>normal operation. GPDMA1 channel 2 continues to operate while CPU is in debug mode.</td>
<td>stop in debug. GPDMA1 channel 2 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>1</td>
<td><strong>DBG_GPDMA1_1_STOP</strong>: GPDMA1 channel 1 stop in debug</td>
<td>normal operation. GPDMA1 channel 1 continues to operate while CPU is in debug mode.</td>
<td>stop in debug. GPDMA1 channel 1 is frozen while CPU is in debug mode.</td>
</tr>
<tr>
<td>0</td>
<td><strong>DBG_GPDMA1_0_STOP</strong>: GPDMA1 channel 0 stop in debug</td>
<td>normal operation. GPDMA1 channel 0 continues to operate while CPU is in debug mode.</td>
<td>stop in debug. GPDMA1 channel 0 is frozen while CPU is in debug mode.</td>
</tr>
</tbody>
</table>
DBGMCU status register (DBGMCU_SR)

Address offset: 0xFC
Reset value: 0x0001 XX03

This register is always accessible.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16  **AP_ENABLED[15:0]**: Bit n identifies whether access port AP n is open (can be accessed via the debug port) or locked (debug access to the AP is blocked)
- Bit n = 0: APn locked
- Bit n = 1: APn enabled

Bits 15:0  **AP_PRESENT[15:0]**: Bit n identifies whether access port AP n is present in device
- Bit n = 0: APn absent
- Bit n = 1: APn present

DBGMCU debug authentication mailbox host register (DBGMCU_DBG_AUTH_HOST)

Address offset: 0x100
Reset value: 0xXXXX XXXX

This register is read only when accessed by the CPU, writes have no effect.

This register can be written and read by an external debugger when system reset is asserted, or when access is granted during the authentication process.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **MESSAGE[31:0]**: Debug host to device mailbox message.
The debug host requests authentication by writing a value to this register prior to releasing the system reset. During debug authentication the debug host communicates with the device (CPU) via this register.
### DBGMCU debug authentication mailbox device register

**(DBGMCU_DBG_AUTH_DEVICE)**

Address offset: 0x104

Reset value: 0xXXXX XXXX

This register is read only when accessed via the debug port, writes have no effect.

This register can be written and read by the CPU.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **MESSAGE[31:0]**: Device to debug host mailbox message.

During debug authentication the device (CPU) communicates with the debug host via this register.

### DBGMCU debug authentication mailbox acknowledge register

**(DBGMCU_DBG_AUTH_ACK)**

Address offset: 0x108

Reset value: 0x0000 0000

This register is always accessible.
Bits 31:2  Reserved, must be kept at reset value.

Bit 1  **DEV_ACK**: Device to host acknowledge.
This bit is set by hardware when the device (CPU) writes a message in the DBGMCU_DBG_AUTH_DEVICE register. It is reset automatically when the host (debugger) reads the message.
0: DBGMCU_DBG_AUTH_DEVICE register is empty
1: DBGMCU_DBG_AUTH_DEVICE register contains an unread message

Bit 0  **HOST_ACK**: Host to device acknowledge.
This bit is set by hardware when the host (debugger) writes a message in the DBGMCU_DBG_AUTH_HOST register. It is reset automatically when the device (CPU) reads the message.
0: DBGMCU_DBG_AUTH_HOST register is empty
1: DBGMCU_DBG_AUTH_HOST register contains an unread message

**DBGMCU CoreSight peripheral identity register 4 (DBGMCU_PIDR4)**
Address offset: 0xFD0
Reset value: 0x0000 0000
This register is always accessible.

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:8**  Reserved, must be kept at reset value.

**Bits 7:4**  **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

**Bits 3:0**  **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC code

**DBGMCU CoreSight peripheral identity register 0 (DBGMCU_PIDR0)**
Address offset: 0xFE0
Reset value: 0x0000 0000
This register is always accessible.

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:8**  Reserved, must be kept at reset value.

**Bits 7:4**  **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

**Bits 3:0**  **PARTNUM[7:0]**: part number
0x0: STMicroelectronics JEDEC code
DBGMCU CoreSight peripheral identity register 1 (DBGMCU_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 0000
This register is always accessible.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x00: DBGMCU part number

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x0: DBGMCU part number

DBGMCU CoreSight peripheral identity register 2 (DBGMCU_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 000A
This register is always accessible.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

0x2: STMicroelectronics JEDEC code
**DBGMCU CoreSight peripheral identity register 3 (DBGMCU_PIDR3)**

Address offset: 0xFEC

Reset value: 0x0000 0000

This register is always accessible.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version

0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified

0x0: no customer modifications

**DBGMCU CoreSight component identity register 0 (DBGMCU_CIDR0)**

Address offset: 0xFF0

Reset value: 0x0000 000D

This register is always accessible.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]

0x0D: common identification value
**DBGMCU CoreSight component identity register 1 (DBGMCU_CIDR1)**

Address offset: 0xFF4

Reset value: 0x0000 00F0

This register is always accessible.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

| Bits 31:8 | Reserved, must be kept at reset value. |

| Bits 7:4 | **Class[3:0]** | component identification bits [15:12] - component class |
|          | 0xF: Non-CoreSight component |

|          | 0x0: common identification value |

**DBGMCU CoreSight component identity register 2 (DBGMCU_CIDR2)**

Address offset: 0xFF8

Reset value: 0x0000 0005

This register is always accessible.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

| Bits 31:8 | Reserved, must be kept at reset value. |

| Bits 7:0 | **Preamble[19:12]** | component identification bits [23:16] |
|          | 0x05: common identification value |
## DBGMCU CoreSight component identity register 3 (DBGMCU_CIDR3)

Address offset: 0xFFC  
Reset value: 0x0000 00B1  
This register is always accessible.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
### Table 716. DBGMCU register map and reset values (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10</td>
<td>DBGMCU_APB2FZR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>DBGMCU_APB3FZR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>DBGMCU_AHB1FZR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>DBGMCU_DBG_AUTH_HOST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>DBGMCU_DBG_AUTH_DEVICE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>DBGMCU_DBG_AUTH_ACK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xF0</td>
<td>DBGMCU_PIDR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xF8</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD0</td>
<td>DBGMCU_PIDR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD4</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 716. DBGMCU register map and reset values (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFE4</td>
<td>DBGMCU_PIDR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>0xFE8</td>
<td>DBGMCU_PIDR2</td>
<td></td>
<td></td>
<td>00000001</td>
<td>01010001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>0xFEC</td>
<td>DBGMCU_PIDR3</td>
<td>JEP106ID[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>0xFF0</td>
<td>DBGMCU_CIDR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00001010</td>
</tr>
<tr>
<td>0xFF4</td>
<td>DBGMCU_CIDR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000001</td>
</tr>
<tr>
<td>0xFF8</td>
<td>DBGMCU_CIDR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000001</td>
</tr>
<tr>
<td>0xFFC</td>
<td>DBGMCU_CIDR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000001</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for register boundary addresses.

58.13 References

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C, 8th Aug 2013
5. 100230_0002_00_en - Arm® Cortex®-M33 Processor r0p2 Technical Reference Manual, Issue 0002-00, 10 May 2017
6. 100232_0001_00_en - Arm® CoreSight™ ETM-M33 r0p1 Technical Reference Manual, Issue 0001-00, 3 February 2017
59 **Device electronic signature**

The device electronic signature is stored in the system memory area of the Flash memory module and can be read using the debug interface or by the CPU. It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match to the characteristics of the devices.

59.1 **Unique device ID register (96 bits)**

The unique device identifier is ideally suited:

- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as part of the security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits cannot be altered by the user.

Base address: 0x08FF F800

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>UID[31:0]: X and Y coordinates on the wafer</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>f r f r f r f r f r f r r r</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>f r f r f r f r f r f r r r</td>
</tr>
</tbody>
</table>

```plaintext
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
f r f r f r f r f r f r r r
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
f r f r f r f r f r f r r r

Bits 31:0  UID[31:0]: X and Y coordinates on the wafer
```
Address offset: 0x04
Read only = 0xXXXX XXXX where X is factory-programmed

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8 **UID[63:40]**: LOT_NUM[23:0]
Lot number (ASCII encoded)

Bits 7:0 **UID[39:32]**: WAF_NUM[7:0]
Wafer number (8-bit unsigned number)

Address offset: 0x08
Read only = 0xXXXX XXXX where X is factory-programmed

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **UID[95:64]**: LOT_NUM[55:24]
Lot number (ASCII encoded)

59.2 **Flash size data register**

Base address: 0x08FF F80C
Address offset: 0x00
Read only = 0xXXXX where X is factory-programmed

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:0 **FLASH_SIZE[15:0]**: Flash memory size
This field indicates the size of the device Flash memory expressed in Kbytes.
As an example, 0x800 corresponds to 2048 Kbytes.
59.3 Package data register

Base address: 0x08FF F80E
Address offset: 0x00
Read only = 0xXXXX where X is factory-programmed

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKG[4:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **PKG[4:0]**: Package type
- 00000: LQFP64
- 00001: VFQFPN68
- 00010: LQFP100
- 00100: LQFP144
- 00111: LQFP176
- 00011: UFBGA176+25
- 00110: UFBGA169
- 01010: LQFP100 SMPS
- 01100: LQFP144 SMPS
- 01101: LQFP176 SMPS
- 01011: UFBGA176+25 SMPS
- 01110: UFBGA169 SMPS
- Others: reserved
Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST’s customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on www.st.com for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.

- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.

- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.

- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.

- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.
61 Revision history

Table 717. Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>28-Feb-2023</td>
<td>1</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>
## Index

### A

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_AWD2CR</td>
<td>1121</td>
</tr>
<tr>
<td>ADC_AWD3CR</td>
<td>1121</td>
</tr>
<tr>
<td>ADC_CALFACT</td>
<td>1123</td>
</tr>
<tr>
<td>ADC_CCR</td>
<td>1126</td>
</tr>
<tr>
<td>ADC_CDR</td>
<td>1129</td>
</tr>
<tr>
<td>ADC_CFGR</td>
<td>1102</td>
</tr>
<tr>
<td>ADC_CFGR2</td>
<td>1106</td>
</tr>
<tr>
<td>ADC_CR</td>
<td>1098</td>
</tr>
<tr>
<td>ADC_CSR</td>
<td>1124</td>
</tr>
<tr>
<td>ADC_DIFSEL</td>
<td>1122</td>
</tr>
<tr>
<td>ADC_DR</td>
<td>1116</td>
</tr>
<tr>
<td>ADC_HWCFGR0</td>
<td>1129</td>
</tr>
<tr>
<td>ADC_IER</td>
<td>1096</td>
</tr>
<tr>
<td>ADC_IPDR</td>
<td>1130</td>
</tr>
<tr>
<td>ADC_ISR</td>
<td>1094</td>
</tr>
<tr>
<td>ADC_JDRy</td>
<td>1120</td>
</tr>
<tr>
<td>ADC_JSR</td>
<td>1116</td>
</tr>
<tr>
<td>ADC_OFRy</td>
<td>1119</td>
</tr>
<tr>
<td>ADC_OR</td>
<td>1123</td>
</tr>
<tr>
<td>ADC_SIDR</td>
<td>1131</td>
</tr>
<tr>
<td>ADC_SMPR1</td>
<td>1109</td>
</tr>
<tr>
<td>ADC_SMPR2</td>
<td>1109</td>
</tr>
<tr>
<td>ADC_SQR1</td>
<td>1112</td>
</tr>
<tr>
<td>ADC_SQR2</td>
<td>1113</td>
</tr>
<tr>
<td>ADC_SQR3</td>
<td>1114</td>
</tr>
<tr>
<td>ADC_SQR4</td>
<td>1115</td>
</tr>
<tr>
<td>ADC_TR1</td>
<td>1110</td>
</tr>
<tr>
<td>ADC_TR2</td>
<td>1111</td>
</tr>
<tr>
<td>ADC_TR3</td>
<td>1112</td>
</tr>
<tr>
<td>ADC_VERR</td>
<td>1130</td>
</tr>
<tr>
<td>AES_CR</td>
<td>1282</td>
</tr>
<tr>
<td>AES_DINR</td>
<td>1285</td>
</tr>
<tr>
<td>AES_DOUTR</td>
<td>1286</td>
</tr>
<tr>
<td>AES_ICR</td>
<td>1293</td>
</tr>
<tr>
<td>AES_IER</td>
<td>1291</td>
</tr>
<tr>
<td>AES_ISR</td>
<td>1292</td>
</tr>
<tr>
<td>AES_IVR0</td>
<td>1288</td>
</tr>
<tr>
<td>AES_IVR1</td>
<td>1288</td>
</tr>
<tr>
<td>AES_IVR2</td>
<td>1288</td>
</tr>
<tr>
<td>AES_IVR3</td>
<td>1289</td>
</tr>
<tr>
<td>AES_KEYR0</td>
<td>1286</td>
</tr>
<tr>
<td>AES_KEYR1</td>
<td>1287</td>
</tr>
<tr>
<td>AES_KEYR2</td>
<td>1287</td>
</tr>
<tr>
<td>AES_KEYR3</td>
<td>1287</td>
</tr>
<tr>
<td>AES_KEYR4</td>
<td>1289</td>
</tr>
<tr>
<td>AES_KEYR5</td>
<td>1289</td>
</tr>
<tr>
<td>AES_KEYR6</td>
<td>1290</td>
</tr>
<tr>
<td>AES_KEYR7</td>
<td>1290</td>
</tr>
<tr>
<td>AES_SR</td>
<td>1284</td>
</tr>
<tr>
<td>AES_SUSPxR</td>
<td>1290</td>
</tr>
<tr>
<td>AP0_CSWR</td>
<td>2963</td>
</tr>
<tr>
<td>AP1_CSWR</td>
<td>2964</td>
</tr>
<tr>
<td>APx_BASER</td>
<td>2967</td>
</tr>
<tr>
<td>APx_BDnR</td>
<td>2966</td>
</tr>
<tr>
<td>APx_DRWR</td>
<td>2966</td>
</tr>
<tr>
<td>APx_IDR</td>
<td>2967</td>
</tr>
<tr>
<td>APx_TAR</td>
<td>2966</td>
</tr>
</tbody>
</table>

### B

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPU_CIDR0</td>
<td>3023</td>
</tr>
<tr>
<td>BPU_CIDR1</td>
<td>3023</td>
</tr>
<tr>
<td>BPU_CIDR2</td>
<td>3024</td>
</tr>
<tr>
<td>BPU_CIDR3</td>
<td>3024</td>
</tr>
<tr>
<td>BPU_COMPxR</td>
<td>3019</td>
</tr>
<tr>
<td>BPU_CTRLR</td>
<td>3018</td>
</tr>
<tr>
<td>BPU_DEVARCHR</td>
<td>3019</td>
</tr>
<tr>
<td>BPU_DEVTYPEPER</td>
<td>3020</td>
</tr>
<tr>
<td>BPU_PIDR0</td>
<td>3021</td>
</tr>
<tr>
<td>BPU_PIDR1</td>
<td>3021</td>
</tr>
<tr>
<td>BPU_PIDR2</td>
<td>3022</td>
</tr>
<tr>
<td>BPU_PIDR3</td>
<td>3022</td>
</tr>
<tr>
<td>BPU_PIDR4</td>
<td>3020</td>
</tr>
</tbody>
</table>

### C

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1ROM_CIDR3</td>
<td>2977, 2983, 2988</td>
</tr>
<tr>
<td>CORDIC_CSR</td>
<td>764</td>
</tr>
<tr>
<td>CORDIC_RDATA</td>
<td>767</td>
</tr>
<tr>
<td>CORDIC_WDATA</td>
<td>766</td>
</tr>
<tr>
<td>CPUROM_CIDR0</td>
<td>2987</td>
</tr>
<tr>
<td>CPUROM_CIDR1</td>
<td>2987</td>
</tr>
<tr>
<td>CPUROM_CIDR2</td>
<td>2988</td>
</tr>
<tr>
<td>CPUROM_CIDR3</td>
<td>2988</td>
</tr>
<tr>
<td>CPUROM_MEMTYPEPER</td>
<td>2984</td>
</tr>
<tr>
<td>CPUROM_PIDR0</td>
<td>2985</td>
</tr>
<tr>
<td>CPUROM_PIDR1</td>
<td>2985</td>
</tr>
<tr>
<td>CPUROM_PIDR2</td>
<td>2986</td>
</tr>
<tr>
<td>CPUROM_PIDR3</td>
<td>2986</td>
</tr>
<tr>
<td>CPUROM_PIDR4</td>
<td>2984</td>
</tr>
<tr>
<td>CRC_CR</td>
<td>747</td>
</tr>
<tr>
<td>CRC_DR</td>
<td>746</td>
</tr>
<tr>
<td>CRC_IDR</td>
<td>746</td>
</tr>
<tr>
<td>CRC_INIT</td>
<td>748</td>
</tr>
<tr>
<td>CRC_POL</td>
<td>748</td>
</tr>
<tr>
<td>CRS_CCFG</td>
<td>556</td>
</tr>
<tr>
<td>Index</td>
<td>RM0481</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>CRS_CR</td>
<td>555</td>
</tr>
<tr>
<td>CRS_ICR</td>
<td>559</td>
</tr>
<tr>
<td>CRS_ISR</td>
<td>557</td>
</tr>
<tr>
<td>CTI_APPCLEAR</td>
<td>3071</td>
</tr>
<tr>
<td>CTI_APPPULSER</td>
<td>3071</td>
</tr>
<tr>
<td>CTI_APPSETR</td>
<td>3070</td>
</tr>
<tr>
<td>CTI_CHINSTSR</td>
<td>3074</td>
</tr>
<tr>
<td>CTI_CHOUTSTSR</td>
<td>3074</td>
</tr>
<tr>
<td>CTI_CIDR0</td>
<td>3079</td>
</tr>
<tr>
<td>CTI_CIDR1</td>
<td>3079</td>
</tr>
<tr>
<td>CTI_CIDR2</td>
<td>3080</td>
</tr>
<tr>
<td>CTI_CIDR3</td>
<td>3080</td>
</tr>
<tr>
<td>CTI_CONTROLR</td>
<td>3069</td>
</tr>
<tr>
<td>CTI_DEVIDR</td>
<td>3075</td>
</tr>
<tr>
<td>CTI_DEVTYPEPER</td>
<td>3076</td>
</tr>
<tr>
<td>CTI_GATER</td>
<td>3075</td>
</tr>
<tr>
<td>CTI_INENxR</td>
<td>3072</td>
</tr>
<tr>
<td>CTI_INTACKR</td>
<td>3069</td>
</tr>
<tr>
<td>CTI_OUTENxR</td>
<td>3072</td>
</tr>
<tr>
<td>CTI_PIDR0</td>
<td>3077</td>
</tr>
<tr>
<td>CTI_PIDR1</td>
<td>3077</td>
</tr>
<tr>
<td>CTI_PIDR2</td>
<td>3078</td>
</tr>
<tr>
<td>CTI_PIDR3</td>
<td>3078</td>
</tr>
<tr>
<td>CTI_PIDR4</td>
<td>3076</td>
</tr>
<tr>
<td>CTI_TRGOSTSR</td>
<td>3073</td>
</tr>
<tr>
<td>CTI_TRGOSTSR</td>
<td>3073</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
<tr>
<td>DAC_CCR</td>
<td>1185</td>
</tr>
<tr>
<td>DAC_CR</td>
<td>1174</td>
</tr>
<tr>
<td>DAC_DHR12L1</td>
<td>1179</td>
</tr>
<tr>
<td>DAC_DHR12L2</td>
<td>1180</td>
</tr>
<tr>
<td>DAC_DHR12LD</td>
<td>1182</td>
</tr>
<tr>
<td>DAC_DHR12R1</td>
<td>1178</td>
</tr>
<tr>
<td>DAC_DHR12R2</td>
<td>1180</td>
</tr>
<tr>
<td>DAC_DHR12RD</td>
<td>1181</td>
</tr>
<tr>
<td>DAC_DHR8R1</td>
<td>1179</td>
</tr>
<tr>
<td>DAC_DHR8R2</td>
<td>1181</td>
</tr>
<tr>
<td>DAC_DHR8RD</td>
<td>1182</td>
</tr>
<tr>
<td>DAC_DOR1</td>
<td>1183</td>
</tr>
<tr>
<td>DAC_DOR2</td>
<td>1183</td>
</tr>
<tr>
<td>DAC_MCR</td>
<td>1186</td>
</tr>
<tr>
<td>DAC_SSHHR</td>
<td>1188</td>
</tr>
<tr>
<td>DAC_SHR</td>
<td>1189</td>
</tr>
<tr>
<td>DAC_SHSR1</td>
<td>1187</td>
</tr>
<tr>
<td>DAC_SHSR2</td>
<td>1188</td>
</tr>
<tr>
<td>DAC_SR</td>
<td>1184</td>
</tr>
<tr>
<td>DAC_SWSWTRGR</td>
<td>1177</td>
</tr>
<tr>
<td>DBGMCU_AHB1FZR</td>
<td>3091</td>
</tr>
<tr>
<td>DBGMCU_APB1HFZR</td>
<td>3088</td>
</tr>
<tr>
<td>DBGMCU_APB1LFZR</td>
<td>3086</td>
</tr>
</tbody>
</table>

3108/3119  RM0481 Rev 1
<table>
<thead>
<tr>
<th>RM0481</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETH_DMACARXBR</td>
<td>2826</td>
</tr>
<tr>
<td>ETH_DMACARXDR</td>
<td>2825</td>
</tr>
<tr>
<td>ETH_DMACATXBR</td>
<td>2826</td>
</tr>
<tr>
<td>ETH_DMACATXDR</td>
<td>2825</td>
</tr>
<tr>
<td>ETH_DMACCR</td>
<td>2811</td>
</tr>
<tr>
<td>ETH_DMACIER</td>
<td>2820</td>
</tr>
<tr>
<td>ETH_DMACMFCR</td>
<td>2829</td>
</tr>
<tr>
<td>ETH_DMACRXCR</td>
<td>2814</td>
</tr>
<tr>
<td>ETH_DMACRXDLAR</td>
<td>2817</td>
</tr>
<tr>
<td>ETH_DMACRXDTPR</td>
<td>2819</td>
</tr>
<tr>
<td>ETH_DMACRXIWR</td>
<td>2824</td>
</tr>
<tr>
<td>ETH_DMACRXLR</td>
<td>2820</td>
</tr>
<tr>
<td>ETH_DMACSR</td>
<td>2826</td>
</tr>
<tr>
<td>ETH_DMACTXCR</td>
<td>2812</td>
</tr>
<tr>
<td>ETH_DMACTXDLAR</td>
<td>2816</td>
</tr>
<tr>
<td>ETH_DMACTXDTPR</td>
<td>2818</td>
</tr>
<tr>
<td>ETH_DMADSR</td>
<td>2810</td>
</tr>
<tr>
<td>ETH_DMAISR</td>
<td>2810</td>
</tr>
<tr>
<td>Index</td>
<td>RM0481</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>ETH_MACRXTXSR</td>
<td>2867</td>
</tr>
<tr>
<td>ETH_MACSPI0R</td>
<td>2928</td>
</tr>
<tr>
<td>ETH_MACSPI1R</td>
<td>2929</td>
</tr>
<tr>
<td>ETH_MACSPI2R</td>
<td>2929</td>
</tr>
<tr>
<td>ETH_MACSSIR</td>
<td>2912</td>
</tr>
<tr>
<td>ETH_MACSTNR</td>
<td>2914</td>
</tr>
<tr>
<td>ETH_MACSTNUR</td>
<td>2915</td>
</tr>
<tr>
<td>ETH_MACSTSUR</td>
<td>2915</td>
</tr>
<tr>
<td>ETH_MACTSAR</td>
<td>2916</td>
</tr>
<tr>
<td>ETH_MACTSCR</td>
<td>2910</td>
</tr>
<tr>
<td>ETH_MACTEACR</td>
<td>2921</td>
</tr>
<tr>
<td>ETH_MACTEACRN</td>
<td>2922</td>
</tr>
<tr>
<td>ETH_MACTSIACR</td>
<td>2921</td>
</tr>
<tr>
<td>ETH_MACTSICRN</td>
<td>2922</td>
</tr>
<tr>
<td>ETH_MACTSSR</td>
<td>2917</td>
</tr>
<tr>
<td>ETH_MACTXTSSNR</td>
<td>2918</td>
</tr>
<tr>
<td>ETH_MACTXTSSSR</td>
<td>2919</td>
</tr>
<tr>
<td>ETH_MACVHTR</td>
<td>2858</td>
</tr>
<tr>
<td>ETH_MACVIR</td>
<td>2859</td>
</tr>
<tr>
<td>ETH_MACVR</td>
<td>2874</td>
</tr>
<tr>
<td>ETH_MACVTR</td>
<td>2856</td>
</tr>
<tr>
<td>ETH_MACWTR</td>
<td>2853</td>
</tr>
<tr>
<td>ETH_MMC_CONTROL</td>
<td>2889</td>
</tr>
<tr>
<td>ETH_MMC_RX_INTERRUPT</td>
<td>2890</td>
</tr>
<tr>
<td>ETH_MMC_RX_INTERRUPT_MASK</td>
<td>2893</td>
</tr>
<tr>
<td>ETH_MMC_TX_INTERRUPT</td>
<td>2891</td>
</tr>
<tr>
<td>ETH_MMC_TX_INTERRUPT_MASK</td>
<td>2894</td>
</tr>
<tr>
<td>ETH_MTLISR</td>
<td>2834</td>
</tr>
<tr>
<td>ETH_MTLOMR</td>
<td>2833</td>
</tr>
<tr>
<td>ETH_MTLQICSR</td>
<td>2837</td>
</tr>
<tr>
<td>ETH_MTLRXQDR</td>
<td>2842</td>
</tr>
<tr>
<td>ETH_MTLRXQMPOCR</td>
<td>2841</td>
</tr>
<tr>
<td>ETH_MTLRXQMOCR</td>
<td>2839</td>
</tr>
<tr>
<td>ETH_MTLTXQDR</td>
<td>2836</td>
</tr>
<tr>
<td>ETH_MTLTXQOMR</td>
<td>2834</td>
</tr>
<tr>
<td>ETH_MTLTXQUR</td>
<td>2835</td>
</tr>
<tr>
<td>ETH_RX_ALIGNMENT_ERROR_PACKETS</td>
<td>2897</td>
</tr>
<tr>
<td>ETH_RX_CRC_ERROR_PACKETS</td>
<td>2896</td>
</tr>
<tr>
<td>ETH_RX_LPI_TRAN_CNTR</td>
<td>2899</td>
</tr>
<tr>
<td>ETH_RX_LPI_USEC_CNTR</td>
<td>2899</td>
</tr>
<tr>
<td>ETH_RX_UNICAST_PACKETS_GOOD</td>
<td>2897</td>
</tr>
<tr>
<td>ETH_TX_LPI_TRAN_CNTR</td>
<td>2898</td>
</tr>
<tr>
<td>ETH_TX_LPI_USEC_CNTR</td>
<td>2898</td>
</tr>
<tr>
<td>ETH_TX_MULTIPLE_COLLISION_-GOOD_PACKETS</td>
<td>2895</td>
</tr>
<tr>
<td>ETH_TX_PACKET_COUNT_GOOD</td>
<td>2896</td>
</tr>
<tr>
<td>ETH_TX_SINGLE_COLLISION_GOOD_PACKETS</td>
<td>2895</td>
</tr>
<tr>
<td>ETS</td>
<td>3045</td>
</tr>
<tr>
<td>ETM_AUTHSTATR</td>
<td>3045</td>
</tr>
<tr>
<td>ETM_CCCTLR</td>
<td>3031</td>
</tr>
</tbody>
</table>
EXTI_FTSR2 ................................. 721
EXTI_IMR1 .................................. 738
EXTI_IMR2 .................................. 739
EXTI_LOCKR .................................. 738
EXTI_PRIVCFGR1 .............................. 720
EXTI_PRIVCFGR2 .............................. 727
EXTI_RPR1 .................................. 718
EXTI_RPR2 .................................. 724
EXTI_RTSR1 .................................. 716
EXTI_RTSR2 .................................. 720
EXTI_SECCFGR1 ............................... 719
EXTI_SECCFGR2 ............................... 727
EXTI_SWIER1 .................................. 717
EXTI_SWIER2 .................................. 723
FDCAN_CCCR .................................. 2550
FDCAN_CKDIV .................................. 2576
FDCAN_CREL .................................. 2547
FDCAN_DBTP .................................. 2548
FDCAN_ENDN .................................. 2547
FDCAN_HPM3 .................................. 2567
FDCAN_IE ..................................... 2561
FDCAN_IIE ..................................... 2564
FDCAN_ILS .................................... 2563
FDCAN_IR ...................................... 2559
FDCAN_NBTP .................................... 2552
FDCAN_PSR .................................... 2556
FDCAN_RWD ..................................... 2550
FDCAN_RXF0A .................................. 2568
FDCAN_RXF0S .................................. 2567
FDCAN_RXF1A .................................. 2569
FDCAN_RXF1S .................................. 2568
FDCAN_RXGC .................................. 2565
FDCAN_TDCR .................................. 2558
FDCAN_TEST ................................... 2549
FDCAN_TOCC ................................... 2554
FDCAN_TOCV ................................... 2555
FDCAN_TSCC ................................... 2553
FDCAN_TSCV ................................... 2554
FDCAN_TXBAR .................................. 2572
FDCAN_TXBC ................................... 2570
FDCAN_TXBCF .................................. 2573
FDCAN_TXBCIE .................................. 2574
FDCAN_TXBCR .................................. 2572
FDCAN_TXBRP .................................. 2571
FDCAN_TXBIE .................................. 2574
FDCAN_TXBTO .................................. 2573
FDCAN_TXEFA .................................. 2575
FDCAN_TXEFS .................................. 2575

FDCAN_TXFS .................................. 2570
FDCAN_XIDAM .................................. 2566
FLASH_ACR ..................................... 305
FLASH_BOOTR_PRG .............................. 334
FLASH_ECCCORR ............................... 341
FLASH_ECCDETR ............................... 342
FLASH_ECCDR ................................. 343
FLASH_EDATA1R_PRG ............................ 339
FLASH_EDATA2R_PRG ............................ 346
FLASH_EDATA2R_PRG ............................ 347
FLASH_HDP1R_CUR ............................. 340
FLASH_HDP1R_PRG ............................. 340
FLASH_HDP2R_CUR ............................. 348
FLASH_HDP2R_PRG ............................. 348
FLASH_HDPEXTR ............................... 325
FLASH_NSBOOTR_CUR ........................... 332
FLASH_NSBOOTR_PRG ........................... 332
FLASH_NSCCR .................................. 320
FLASH_NSCCR .................................. 314
FLASH_NSEPOCHR_CUR .......................... 329
FLASH_NSKEYR .................................. 306
FLASH_NSOKCFGR ................................ 322
FLASH_NSOKCKEYR ................................ 307
FLASH_NSPOCHR ................................ 310
FLASH_OPSR .................................... 308
FLASH_OPTCR .................................... 309
FLASH_OPTKEYR ................................ 307
FLASH_OPTSR_CUR ............................... 325
FLASH_OPSR_PRG ................................ 327
FLASH_OPSR2_CUR ............................... 330
FLASH_OPSR2_PRG ................................ 331
FLASH_OTPBLR_CUR .............................. 334
FLASH_OTPBLR_PRG .............................. 335
FLASH_PRIVBB1Rx ................................ 336
FLASH_PRIVBB2Rx ................................ 344
FLASH_PRIVCFFGR ............................... 322
FLASH_SECBB1Rx ................................ 335
FLASH_SECBB2Rx ................................ 343
FLASH_SECBOOTR_CUR .......................... 333
FLASH_SECCCR .................................. 321
FLASH_SECCR .................................... 317
FLASH_SECEPOCHR_CUR .......................... 329
FLASH_SECKEYR ................................ 306
FLASH_SECBBKCFGR ............................. 324
FLASH_SECBBKKEYR ............................. 308
FLASH_SECCR .................................... 312
FLASH_SECWM1R_CUR ........................... 336
FLASH_SECWM1R_PRG ........................... 337
FLASH_SECWM2R_CUR ........................... 344
FLASH_SECWM2R_PRG ........................... 345
FLASH_WRP1R_PRG ............................. 337
<table>
<thead>
<tr>
<th>Index</th>
<th>RM0481</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLASH_WRP1R_PRG</td>
<td>.338</td>
</tr>
<tr>
<td>FLASH_WRP2R_CUR</td>
<td>.345</td>
</tr>
<tr>
<td>FLASH_WRP2R_PRG</td>
<td>.346</td>
</tr>
<tr>
<td>FMAC_CR</td>
<td>.791</td>
</tr>
<tr>
<td>FMAC_PARAM</td>
<td>.790</td>
</tr>
<tr>
<td>FMAC_RDATA</td>
<td>.794</td>
</tr>
<tr>
<td>FMAC_SR</td>
<td>.792</td>
</tr>
<tr>
<td>FMAC_WDATA</td>
<td>.793</td>
</tr>
<tr>
<td>FMAC_X1BUFCFG</td>
<td>.788</td>
</tr>
<tr>
<td>FMAC_X2BUFCFG</td>
<td>.788</td>
</tr>
<tr>
<td>FMAC_YBUF_CFG</td>
<td>.789</td>
</tr>
<tr>
<td>GPDMA_CxBR1</td>
<td>.686-687</td>
</tr>
<tr>
<td>GPDMA_CxBR2</td>
<td>.694</td>
</tr>
<tr>
<td>GPDMA_CxCR</td>
<td>.676</td>
</tr>
<tr>
<td>GPDMA_CxDAR</td>
<td>.692</td>
</tr>
<tr>
<td>GPDMA_CxFCR</td>
<td>.673</td>
</tr>
<tr>
<td>GPDMA_CxLBAR</td>
<td>.673</td>
</tr>
<tr>
<td>GPDMA_CxLLR</td>
<td>.695, 697</td>
</tr>
<tr>
<td>GPDMA_CxSAR</td>
<td>.690</td>
</tr>
<tr>
<td>GPDMA_CxSR</td>
<td>.674</td>
</tr>
<tr>
<td>GPDMA_CxTR1</td>
<td>.678</td>
</tr>
<tr>
<td>GPDMA_CxTR2</td>
<td>.682</td>
</tr>
<tr>
<td>GPDMA_CxTR3</td>
<td>.693</td>
</tr>
<tr>
<td>GPDMA_MISR</td>
<td>.671</td>
</tr>
<tr>
<td>GPDMA_PRIVCFG</td>
<td>.670</td>
</tr>
<tr>
<td>GPDMA_RCFGLOCKR</td>
<td>.670</td>
</tr>
<tr>
<td>GPDMA_SECCFG</td>
<td>.669</td>
</tr>
<tr>
<td>GPDMA_SMIISR</td>
<td>.672</td>
</tr>
<tr>
<td>GPIOx_AFRH</td>
<td>.576</td>
</tr>
<tr>
<td>GPIOx_AFRL</td>
<td>.575</td>
</tr>
<tr>
<td>GPIOx_BRR</td>
<td>.577</td>
</tr>
<tr>
<td>GPIOx_BSRR</td>
<td>.574</td>
</tr>
<tr>
<td>GPIOx_HSLVR</td>
<td>.578</td>
</tr>
<tr>
<td>GPIOx_IDR</td>
<td>.573</td>
</tr>
<tr>
<td>GPIOx_LCKR</td>
<td>.574</td>
</tr>
<tr>
<td>GPIOx_MODER</td>
<td>.571</td>
</tr>
<tr>
<td>GPIOx_ODR</td>
<td>.573</td>
</tr>
<tr>
<td>GPIOx_OSPEDR</td>
<td>.572</td>
</tr>
<tr>
<td>GPIOx_OTYPER</td>
<td>.571</td>
</tr>
<tr>
<td>GPIOx_PUPDR</td>
<td>.572</td>
</tr>
<tr>
<td>GPIOx_SECCFG</td>
<td>.578</td>
</tr>
<tr>
<td>GTZC1_MPCBBz_CFGLOCK1</td>
<td>.220</td>
</tr>
<tr>
<td>GTZC1_MPCBBz_CR</td>
<td>.219</td>
</tr>
<tr>
<td>GTZC1_MPCBBz_PRIVCFG</td>
<td>.221</td>
</tr>
<tr>
<td>GTZC1_MPCBBz_SECCFG</td>
<td>.220</td>
</tr>
<tr>
<td>GTZC1_TZIC_FCR1</td>
<td>.208</td>
</tr>
<tr>
<td>GTZC1_TZIC_FCR2</td>
<td>.210</td>
</tr>
<tr>
<td>GTZC1_TZIC_FCR3</td>
<td>.212</td>
</tr>
<tr>
<td>GTZC1_TZIC_FCR4</td>
<td>.214</td>
</tr>
<tr>
<td>GTZC1_TZIC_IER1</td>
<td>.191</td>
</tr>
<tr>
<td>GTZC1_TZIC_IER2</td>
<td>.193</td>
</tr>
<tr>
<td>GTZC1_TZIC_IER3</td>
<td>.195</td>
</tr>
<tr>
<td>GTZC1_TZIC_IER4</td>
<td>.197</td>
</tr>
<tr>
<td>GTZC1_TZIC_SR1</td>
<td>.199</td>
</tr>
<tr>
<td>GTZC1_TZIC_SR2</td>
<td>.202</td>
</tr>
<tr>
<td>GTZC1_TZIC_SR3</td>
<td>.204</td>
</tr>
<tr>
<td>GTZC1_TZIC_SR4</td>
<td>.206</td>
</tr>
<tr>
<td>GTZC1_TZSC_CR</td>
<td>.172</td>
</tr>
<tr>
<td>GTZC1_TZSC_MPCWMxAR</td>
<td>.187</td>
</tr>
<tr>
<td>GTZC1_TZSC_MPCWMxBR</td>
<td>.188</td>
</tr>
<tr>
<td>GTZC1_TZSC_MPCWMxCFG</td>
<td>.186</td>
</tr>
<tr>
<td>GTZC1_TZSC_PRIVCFG1</td>
<td>.179</td>
</tr>
<tr>
<td>GTZC1_TZSC_PRIVCFG2</td>
<td>.181</td>
</tr>
<tr>
<td>GTZC1_TZSC_PRIVCFG3</td>
<td>.184</td>
</tr>
<tr>
<td>GTZC1_TZSC_SECCFG1</td>
<td>.172</td>
</tr>
<tr>
<td>GTZC1_TZSC_SECCFG2</td>
<td>.175</td>
</tr>
<tr>
<td>GTZC1_TZSC_SECCFG3</td>
<td>.177</td>
</tr>
<tr>
<td>HASH_CR</td>
<td>.1362</td>
</tr>
<tr>
<td>HASH_CSR</td>
<td>.1370</td>
</tr>
<tr>
<td>HASH_DIN</td>
<td>.1364</td>
</tr>
<tr>
<td>HASH_HRAx</td>
<td>.1367</td>
</tr>
<tr>
<td>HASH_HRx</td>
<td>.1367</td>
</tr>
<tr>
<td>HASH_IMR</td>
<td>.1368</td>
</tr>
<tr>
<td>HASH_SR</td>
<td>.1368</td>
</tr>
<tr>
<td>HASH_STR</td>
<td>.1365</td>
</tr>
<tr>
<td>I2C_CR1</td>
<td>.2103</td>
</tr>
<tr>
<td>I2C_CR2</td>
<td>.2106</td>
</tr>
<tr>
<td>I2C_ICR</td>
<td>.2114</td>
</tr>
<tr>
<td>I2C_ISR</td>
<td>.2112</td>
</tr>
<tr>
<td>I2C_OAR1</td>
<td>.2108</td>
</tr>
<tr>
<td>I2C_OAR2</td>
<td>.2109</td>
</tr>
<tr>
<td>I2C_PECR</td>
<td>.2115</td>
</tr>
<tr>
<td>Index</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>I2C_RXDR</td>
<td>2116</td>
</tr>
<tr>
<td>I2C_TIMEOUTR</td>
<td>2111</td>
</tr>
<tr>
<td>I2C_TIMINGR</td>
<td>2110</td>
</tr>
<tr>
<td>I2C_TXDR</td>
<td>2116</td>
</tr>
<tr>
<td>I3C_BCR</td>
<td>2218</td>
</tr>
<tr>
<td>I3C_CEVR</td>
<td>2206</td>
</tr>
<tr>
<td>I3C_CFGPR</td>
<td>2185</td>
</tr>
<tr>
<td>I3C_CR</td>
<td>2183, 2181</td>
</tr>
<tr>
<td>I3C_CRCAPR</td>
<td>2221</td>
</tr>
<tr>
<td>I3C_DCR</td>
<td>2219</td>
</tr>
<tr>
<td>I3C_DEVR0</td>
<td>2208</td>
</tr>
<tr>
<td>I3C_DEVRx</td>
<td>2210</td>
</tr>
<tr>
<td>I3C_EPIDR</td>
<td>2224</td>
</tr>
<tr>
<td>I3C_EVR</td>
<td>2200</td>
</tr>
<tr>
<td>I3C_GETCAPR</td>
<td>2220</td>
</tr>
<tr>
<td>I3C_GETMXDSR</td>
<td>2222</td>
</tr>
<tr>
<td>I3C_IBIDR</td>
<td>2194</td>
</tr>
<tr>
<td>I3C_IER</td>
<td>2204</td>
</tr>
<tr>
<td>I3C_MAXRLR</td>
<td>2212</td>
</tr>
<tr>
<td>I3C_MAXWLR</td>
<td>2213</td>
</tr>
<tr>
<td>I3C_RDR</td>
<td>2190</td>
</tr>
<tr>
<td>I3C_RDWR</td>
<td>2190</td>
</tr>
<tr>
<td>I3C_RMR</td>
<td>2199</td>
</tr>
<tr>
<td>I3C_SER</td>
<td>2197</td>
</tr>
<tr>
<td>I3C_SR</td>
<td>2196</td>
</tr>
<tr>
<td>I3C_TDR</td>
<td>2191</td>
</tr>
<tr>
<td>I3C_TDWR</td>
<td>2192</td>
</tr>
<tr>
<td>I3C_TGTDR</td>
<td>2196</td>
</tr>
<tr>
<td>I3C_TIMINGR0</td>
<td>2214</td>
</tr>
<tr>
<td>I3C_TIMINGR1</td>
<td>2215</td>
</tr>
<tr>
<td>I3C_TIMINGR2</td>
<td>2217</td>
</tr>
<tr>
<td>ICACHE_CCR</td>
<td>366</td>
</tr>
<tr>
<td>ICACHE_CRRx</td>
<td>369</td>
</tr>
<tr>
<td>ICACHE_FCR</td>
<td>368</td>
</tr>
<tr>
<td>ICACHE_HMONR</td>
<td>369</td>
</tr>
<tr>
<td>ICACHE_IER</td>
<td>368</td>
</tr>
<tr>
<td>ICACHE_MMONR</td>
<td>369</td>
</tr>
<tr>
<td>ICACHE_SR</td>
<td>367</td>
</tr>
<tr>
<td>ITM_CIDR0</td>
<td>3015</td>
</tr>
<tr>
<td>ITM_CIDR1</td>
<td>3015</td>
</tr>
<tr>
<td>ITM_CIDR2</td>
<td>3016</td>
</tr>
<tr>
<td>ITM_CIDR3</td>
<td>3016</td>
</tr>
<tr>
<td>ITM_DEVARCHR</td>
<td>3011</td>
</tr>
<tr>
<td>ITM_DEVTYPEPER</td>
<td>3012</td>
</tr>
<tr>
<td>ITM_PIDR0</td>
<td>3013</td>
</tr>
<tr>
<td>ITM_PIDR1</td>
<td>3013</td>
</tr>
<tr>
<td>ITM_PIDR2</td>
<td>3014</td>
</tr>
<tr>
<td>ITM_PIDR3</td>
<td>3014</td>
</tr>
<tr>
<td>ITM_PIDR4</td>
<td>3012</td>
</tr>
<tr>
<td>ITM_STIMRx</td>
<td>3008</td>
</tr>
<tr>
<td>ITM_TCR</td>
<td>3010</td>
</tr>
<tr>
<td>ITM_TER</td>
<td>3009</td>
</tr>
<tr>
<td>ITM_TPR</td>
<td>3009</td>
</tr>
<tr>
<td>IWDG_EWCR</td>
<td>1937</td>
</tr>
<tr>
<td>IWDG_KR</td>
<td>1934</td>
</tr>
<tr>
<td>IWDG_PR</td>
<td>1934</td>
</tr>
<tr>
<td>IWDG_RLR</td>
<td>1935</td>
</tr>
<tr>
<td>IWDG_SR</td>
<td>1935</td>
</tr>
<tr>
<td>IWDG_WINR</td>
<td>1937</td>
</tr>
<tr>
<td>ITM_TPR</td>
<td>3009</td>
</tr>
<tr>
<td>LPTIM_ARR</td>
<td>1917</td>
</tr>
<tr>
<td>LPTIM_CCMR1</td>
<td>1919</td>
</tr>
<tr>
<td>LPTIM_CCR1</td>
<td>1916</td>
</tr>
<tr>
<td>LPTIM_CCR2</td>
<td>1922</td>
</tr>
<tr>
<td>LPTIM_CFGR</td>
<td>1912</td>
</tr>
<tr>
<td>LPTIM_CFGR2</td>
<td>1918</td>
</tr>
<tr>
<td>LPTIM_CNT</td>
<td>1917</td>
</tr>
<tr>
<td>LPTIM_TPR</td>
<td>3009</td>
</tr>
<tr>
<td>LPUART_BRR</td>
<td>2358</td>
</tr>
<tr>
<td>LPUART_CR1</td>
<td>2347, 2350</td>
</tr>
<tr>
<td>LPUART_CR2</td>
<td>2353</td>
</tr>
<tr>
<td>LPUART_CR3</td>
<td>2355</td>
</tr>
<tr>
<td>LPUART_ICR</td>
<td>2367</td>
</tr>
<tr>
<td>LPUART_PRESC</td>
<td>2369</td>
</tr>
<tr>
<td>LPUART_RDR</td>
<td>2368</td>
</tr>
<tr>
<td>LPUART_RQR</td>
<td>2359</td>
</tr>
<tr>
<td>LPUART_TDR</td>
<td>2369</td>
</tr>
<tr>
<td>LPUART_TDR</td>
<td>2369</td>
</tr>
<tr>
<td>LPUART_BRR</td>
<td>2358</td>
</tr>
<tr>
<td>LPUART_CR1</td>
<td>2347, 2350</td>
</tr>
<tr>
<td>LPUART_ICR</td>
<td>2367</td>
</tr>
<tr>
<td>LPUART_ISR</td>
<td>2359, 2364</td>
</tr>
<tr>
<td>LPUART_PRESC</td>
<td>2369</td>
</tr>
<tr>
<td>OCTOSPI_ABR</td>
<td>913</td>
</tr>
<tr>
<td>OCTOSPI_AR</td>
<td>908</td>
</tr>
<tr>
<td>MCUROM_CIDR0</td>
<td>2981</td>
</tr>
<tr>
<td>MCUROM_CIDR1</td>
<td>2981</td>
</tr>
<tr>
<td>MCUROM_CIDR2</td>
<td>2982</td>
</tr>
<tr>
<td>MCUROM_CIDR3</td>
<td>2982</td>
</tr>
<tr>
<td>MCUROM_MEMTYPEPER</td>
<td>2978</td>
</tr>
<tr>
<td>MCUROM_PIDR0</td>
<td>2979</td>
</tr>
<tr>
<td>MCUROM_PIDR1</td>
<td>2980</td>
</tr>
<tr>
<td>MCUROM_PIDR2</td>
<td>2980</td>
</tr>
<tr>
<td>MCUROM_PIDR3</td>
<td>2980</td>
</tr>
<tr>
<td>MCUROM_PIDR4</td>
<td>2979</td>
</tr>
<tr>
<td>OCTOSPI_CCR</td>
<td>OCTOSPI_CR</td>
</tr>
<tr>
<td>----------------</td>
<td>------------</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>OCTOSPI_PIR</td>
<td>OCTOSPI_PSMKRR</td>
</tr>
<tr>
<td>OCTOSPI_IR</td>
<td>OCTOSPI_IR</td>
</tr>
<tr>
<td>OCTOSPI_IR</td>
<td>OCTOSPI_IR</td>
</tr>
<tr>
<td>OCTOSPI_IR</td>
<td>OCTOSPI_IR</td>
</tr>
<tr>
<td>OCTOSPI_IR</td>
<td>OCTOSPI_IR</td>
</tr>
<tr>
<td>OCTOSPI_IR</td>
<td>OCTOSPI_IR</td>
</tr>
<tr>
<td>OCTOSPI_IR</td>
<td>OCTOSPI_IR</td>
</tr>
<tr>
<td>OCTOSPI_IR</td>
<td>OCTOSPI_IR</td>
</tr>
<tr>
<td>OCTOSPI_IR</td>
<td>OCTOSPI_IR</td>
</tr>
<tr>
<td>OCTOSPI_IR</td>
<td>OCTOSPI_IR</td>
</tr>
<tr>
<td>RM0481</td>
<td>Index</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>RCC_BDCR</td>
<td>.536</td>
</tr>
<tr>
<td>RCC_CCIPR1</td>
<td>.528</td>
</tr>
<tr>
<td>RCC_CCIPR2</td>
<td>.530</td>
</tr>
<tr>
<td>RCC_CCIPR3</td>
<td>.532</td>
</tr>
<tr>
<td>RCC_CCIPR4</td>
<td>.534</td>
</tr>
<tr>
<td>RCC_CCIPR5</td>
<td>.535</td>
</tr>
<tr>
<td>RCC_CIFR</td>
<td>.491</td>
</tr>
<tr>
<td>RCC_CIER</td>
<td>.490</td>
</tr>
<tr>
<td>RCC_CICR</td>
<td>.493</td>
</tr>
<tr>
<td>RCC_CIFCR</td>
<td>.473</td>
</tr>
<tr>
<td>RCC_CSICFGR</td>
<td>.473</td>
</tr>
<tr>
<td>RCC_HSICFGR</td>
<td>.472</td>
</tr>
<tr>
<td>RCC_PLL1CFGR</td>
<td>.479</td>
</tr>
<tr>
<td>RCC_PLL1DIVR</td>
<td>.484</td>
</tr>
<tr>
<td>RCC_PLL1FRACR</td>
<td>.485</td>
</tr>
<tr>
<td>RCC_PLL2CFGR</td>
<td>.481</td>
</tr>
<tr>
<td>RCC_PLL2DIVR</td>
<td>.486</td>
</tr>
<tr>
<td>RCC_PLL2FRACR</td>
<td>.487</td>
</tr>
<tr>
<td>RCC_PLL3CFGR</td>
<td>.482</td>
</tr>
<tr>
<td>RCC_PLL3DIVR</td>
<td>.488</td>
</tr>
<tr>
<td>RCC_PLL3FRACR</td>
<td>.489</td>
</tr>
<tr>
<td>RCC_PRIVCFGR</td>
<td>.542</td>
</tr>
<tr>
<td>RCC_RSR</td>
<td>.539</td>
</tr>
<tr>
<td>RCC_SECCFG</td>
<td>.540</td>
</tr>
<tr>
<td>RNG_CR</td>
<td>.1244</td>
</tr>
<tr>
<td>RNG_DR</td>
<td>1248</td>
</tr>
<tr>
<td>RNG_HTCR</td>
<td>1248</td>
</tr>
<tr>
<td>RNG_SR</td>
<td>1247</td>
</tr>
<tr>
<td>RTC_ALRABINR</td>
<td>2000</td>
</tr>
<tr>
<td>RTC_ALRBBINR</td>
<td>2000</td>
</tr>
<tr>
<td>RTC_ALRMAR</td>
<td>1990</td>
</tr>
<tr>
<td>RTC_ALRMASSR</td>
<td>1992</td>
</tr>
<tr>
<td>RTC_ALRMBR</td>
<td>1993</td>
</tr>
<tr>
<td>RTC_ALRMBSSR</td>
<td>1994</td>
</tr>
<tr>
<td>RTC_CALR</td>
<td>1986</td>
</tr>
<tr>
<td>RTC_CR</td>
<td>1978</td>
</tr>
<tr>
<td>RTC_DR</td>
<td>1973</td>
</tr>
<tr>
<td>RTC_ICSR</td>
<td>1975</td>
</tr>
<tr>
<td>RTC_MISR</td>
<td>1996</td>
</tr>
<tr>
<td>RTC_OR</td>
<td>1999</td>
</tr>
<tr>
<td>RTC_PRER</td>
<td>1977</td>
</tr>
<tr>
<td>RTC_PRIVCFGR</td>
<td>1982</td>
</tr>
<tr>
<td>RTC_SCR</td>
<td>1998</td>
</tr>
<tr>
<td>RTC_SECCFG</td>
<td>1984</td>
</tr>
<tr>
<td>RTC_SHIFTR</td>
<td>1987</td>
</tr>
<tr>
<td>RTC_SMISR</td>
<td>1997</td>
</tr>
<tr>
<td>RTC_SR</td>
<td>1995</td>
</tr>
<tr>
<td>RTC_SSR</td>
<td>1974</td>
</tr>
<tr>
<td>RTC_TR</td>
<td>1972</td>
</tr>
<tr>
<td>RTC_SSDR</td>
<td>.1989</td>
</tr>
<tr>
<td>RTC_TSSSR</td>
<td>.1990</td>
</tr>
<tr>
<td>RTC_TSTR</td>
<td>.1988</td>
</tr>
<tr>
<td>RTC_WPR</td>
<td>.1985</td>
</tr>
<tr>
<td>RTC_WUTR</td>
<td>.1978</td>
</tr>
</tbody>
</table>

**S**

<table>
<thead>
<tr>
<th>RM0481</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAEs_CR</td>
<td>.1335</td>
</tr>
<tr>
<td>SAEs_DINR</td>
<td>.1339</td>
</tr>
<tr>
<td>SAEs_DOUTR</td>
<td>.1340</td>
</tr>
<tr>
<td>SAEs_ICR</td>
<td>.1347</td>
</tr>
<tr>
<td>SAEs_IER</td>
<td>.1345</td>
</tr>
<tr>
<td>SAEs_ISR</td>
<td>.1346</td>
</tr>
<tr>
<td>SAEs_IVR0</td>
<td>.1342</td>
</tr>
<tr>
<td>SAEs_IVR1</td>
<td>.1342</td>
</tr>
<tr>
<td>SAEs_IVR2</td>
<td>.1342</td>
</tr>
<tr>
<td>SAEs_IVR3</td>
<td>.1343</td>
</tr>
<tr>
<td>SAEs_KEYR0</td>
<td>.1340</td>
</tr>
<tr>
<td>SAEs_KEYR1</td>
<td>.1341</td>
</tr>
<tr>
<td>SAEs_KEYR2</td>
<td>.1341</td>
</tr>
<tr>
<td>SAEs_KEYR3</td>
<td>.1341</td>
</tr>
<tr>
<td>SAEs_KEYR4</td>
<td>.1343</td>
</tr>
<tr>
<td>SAEs_KEYR5</td>
<td>.1343</td>
</tr>
<tr>
<td>SAEs_KEYR6</td>
<td>.1344</td>
</tr>
<tr>
<td>SAEs_KEYR7</td>
<td>.1344</td>
</tr>
<tr>
<td>SAEs_SR</td>
<td>.1338</td>
</tr>
<tr>
<td>SAEs_SUSPXR</td>
<td>.1344</td>
</tr>
<tr>
<td>SAI_ACLRFR</td>
<td>.2508</td>
</tr>
<tr>
<td>SAI_ACR1</td>
<td>.2487</td>
</tr>
<tr>
<td>SAI_ACR2</td>
<td>.2492</td>
</tr>
<tr>
<td>SAI_ADR</td>
<td>.2510</td>
</tr>
<tr>
<td>SAI_AFRCR</td>
<td>.2496</td>
</tr>
<tr>
<td>SAI_AIM</td>
<td>.2501</td>
</tr>
<tr>
<td>SAI_ASLCTR</td>
<td>.2499</td>
</tr>
<tr>
<td>SAI_ASR</td>
<td>.2504</td>
</tr>
<tr>
<td>SAI_BCLRFR</td>
<td>.2509</td>
</tr>
<tr>
<td>SAI_BCR1</td>
<td>.2489</td>
</tr>
<tr>
<td>SAI_BCR2</td>
<td>.2494</td>
</tr>
<tr>
<td>SAI_BDR</td>
<td>.2511</td>
</tr>
<tr>
<td>SAI_BFSRCR</td>
<td>.2498</td>
</tr>
<tr>
<td>SAI_BIM</td>
<td>.2503</td>
</tr>
<tr>
<td>SAI_BSLCTR</td>
<td>.2500</td>
</tr>
<tr>
<td>SAI_BSR</td>
<td>.2506</td>
</tr>
<tr>
<td>SAI_GCR</td>
<td>.2486</td>
</tr>
<tr>
<td>SAI_PDMCR</td>
<td>.2511</td>
</tr>
<tr>
<td>SAI_PDMDL</td>
<td>.2512</td>
</tr>
<tr>
<td>SBS_CCSCSR</td>
<td>.599</td>
</tr>
<tr>
<td>SBS_CCSCWCR</td>
<td>.601</td>
</tr>
<tr>
<td>SBS_CCVALR</td>
<td>.600</td>
</tr>
<tr>
<td>SBS_CFGR2</td>
<td>.602</td>
</tr>
<tr>
<td>SBS_CNSLCKR</td>
<td>.603</td>
</tr>
<tr>
<td>Index</td>
<td>RM0481</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>SBS_CSLCKR</td>
<td>604</td>
</tr>
<tr>
<td>SBSDBGCR</td>
<td>594</td>
</tr>
<tr>
<td>SBSDBGLOCKR</td>
<td>594</td>
</tr>
<tr>
<td>SBS_ECCNMIR</td>
<td>605</td>
</tr>
<tr>
<td>SBS_EPOCHESELCR</td>
<td>595</td>
</tr>
<tr>
<td>SBS_FPUIMR</td>
<td>598</td>
</tr>
<tr>
<td>SBS_HDPLCR</td>
<td>592</td>
</tr>
<tr>
<td>SBS_HDPLSR</td>
<td>593</td>
</tr>
<tr>
<td>SBS_MESR</td>
<td>599</td>
</tr>
<tr>
<td>SBS_NEXTHDPLCR</td>
<td>597</td>
</tr>
<tr>
<td>SBS_RSSCMDR</td>
<td>595</td>
</tr>
<tr>
<td>SBS_SECCFGR</td>
<td>596</td>
</tr>
<tr>
<td>SDMMC_ACKTIMER</td>
<td>1001</td>
</tr>
<tr>
<td>SDMMC_ARGR</td>
<td>986</td>
</tr>
<tr>
<td>SDMMC_CLKCR</td>
<td>984</td>
</tr>
<tr>
<td>SDMMC_CMDR</td>
<td>986</td>
</tr>
<tr>
<td>SDMMC_DCNTR</td>
<td>992</td>
</tr>
<tr>
<td>SDMMC_DCTRL</td>
<td>991</td>
</tr>
<tr>
<td>SDMMC_DLENR</td>
<td>990</td>
</tr>
<tr>
<td>SDMMC_DTIMER</td>
<td>989</td>
</tr>
<tr>
<td>SDMMC_FIFORx</td>
<td>1001</td>
</tr>
<tr>
<td>SDMMC_ICR</td>
<td>996</td>
</tr>
<tr>
<td>SDMMC_IDMABAR</td>
<td>1004</td>
</tr>
<tr>
<td>SDMMC_IDMABASER</td>
<td>1003</td>
</tr>
<tr>
<td>SDMMC_IDMBSIZER</td>
<td>1002</td>
</tr>
<tr>
<td>SDMMC_IDMACTRLR</td>
<td>1002</td>
</tr>
<tr>
<td>SDMMC_IDMALAR</td>
<td>1003</td>
</tr>
<tr>
<td>SDMMC_MASKR</td>
<td>998</td>
</tr>
<tr>
<td>SDMMC_POWER</td>
<td>983</td>
</tr>
<tr>
<td>SDMMC_RESPCMDR</td>
<td>988</td>
</tr>
<tr>
<td>SDMMC_RESPxR</td>
<td>989</td>
</tr>
<tr>
<td>SDMMC_STAR</td>
<td>993</td>
</tr>
<tr>
<td>SPI_CFG1</td>
<td>2430</td>
</tr>
<tr>
<td>SPI_CFG2</td>
<td>2433</td>
</tr>
<tr>
<td>SPI_CR1</td>
<td>2427</td>
</tr>
<tr>
<td>SPI_CR2</td>
<td>2429</td>
</tr>
<tr>
<td>SPI_CRCPOLY</td>
<td>2441</td>
</tr>
<tr>
<td>SPI_I2SCFGR</td>
<td>2444</td>
</tr>
<tr>
<td>SPI_IER</td>
<td>2436</td>
</tr>
<tr>
<td>SPI_IFCR</td>
<td>2439</td>
</tr>
<tr>
<td>SPI_RXCRC</td>
<td>2443</td>
</tr>
<tr>
<td>SPI_RXDR</td>
<td>2441</td>
</tr>
<tr>
<td>SPI_SR</td>
<td>2437</td>
</tr>
<tr>
<td>SPI_TXCRC</td>
<td>2442</td>
</tr>
<tr>
<td>SPI_TXDR</td>
<td>2440</td>
</tr>
<tr>
<td>SPI_UDRDR</td>
<td>2443</td>
</tr>
<tr>
<td>SYSROM_CIDR0</td>
<td>2976</td>
</tr>
<tr>
<td>SYSROM_CIDR1</td>
<td>2976</td>
</tr>
<tr>
<td>SYSROM_CIDR2</td>
<td>2977</td>
</tr>
<tr>
<td>SYSROM_CIDR3</td>
<td>2977</td>
</tr>
<tr>
<td>SYSROM_MEMTYPEPER</td>
<td>2973</td>
</tr>
<tr>
<td>SYSROM_PIDR0</td>
<td>2974</td>
</tr>
<tr>
<td>SYSROM_PIDR1</td>
<td>2974</td>
</tr>
<tr>
<td>SYSROM_PIDR2</td>
<td>2975</td>
</tr>
<tr>
<td>SYSROM_PIDR3</td>
<td>2975</td>
</tr>
<tr>
<td>SYSROM_PIDR4</td>
<td>2973</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAM_PATCR1</td>
</tr>
<tr>
<td>TAM_PATCR2</td>
</tr>
<tr>
<td>TAM_PATOR</td>
</tr>
<tr>
<td>TAM_ATSEEDR</td>
</tr>
<tr>
<td>TAM_BKPxR</td>
</tr>
<tr>
<td>TAM_COUNT1R</td>
</tr>
<tr>
<td>TAM_CR1</td>
</tr>
<tr>
<td>TAM_CR2</td>
</tr>
<tr>
<td>TAM_CR3</td>
</tr>
<tr>
<td>TAM_FLTCR</td>
</tr>
<tr>
<td>TAM_IER</td>
</tr>
<tr>
<td>TAM_MISR</td>
</tr>
<tr>
<td>TAM_OR</td>
</tr>
<tr>
<td>TAM_PRIVCFGR</td>
</tr>
<tr>
<td>TAM_RPCFGR</td>
</tr>
<tr>
<td>TAM_SCR</td>
</tr>
<tr>
<td>TAM_SECCFGR</td>
</tr>
<tr>
<td>TAM_SMISR</td>
</tr>
<tr>
<td>TAM_SR</td>
</tr>
<tr>
<td>TIM12_ARR</td>
</tr>
<tr>
<td>TIM12_CCR</td>
</tr>
<tr>
<td>TIM12_CCR2</td>
</tr>
<tr>
<td>TIM12_CNT</td>
</tr>
<tr>
<td>TIM12_CR1</td>
</tr>
<tr>
<td>TIM12_CR2</td>
</tr>
<tr>
<td>TIM12_DIER</td>
</tr>
<tr>
<td>TIM12_EGR</td>
</tr>
<tr>
<td>TIM12_PSC</td>
</tr>
<tr>
<td>TIM12_SMCR</td>
</tr>
<tr>
<td>TIM12_SR</td>
</tr>
<tr>
<td>TIM12_TISEL</td>
</tr>
<tr>
<td>TIM15_AF1</td>
</tr>
<tr>
<td>TIM15_AF2</td>
</tr>
<tr>
<td>TIM15_ARR</td>
</tr>
<tr>
<td>TIM15_BDTR</td>
</tr>
<tr>
<td>TIM15_CCER</td>
</tr>
<tr>
<td>TIM15_CCMR1</td>
</tr>
<tr>
<td>TIM15_CCR1</td>
</tr>
<tr>
<td>TIM15_CCR2</td>
</tr>
<tr>
<td>TIM15_CNT</td>
</tr>
<tr>
<td>TIM15_CR1</td>
</tr>
<tr>
<td>TIM15_CR2</td>
</tr>
</tbody>
</table>
TPIU_DEVTYPER .......................... 3061
TPIU_CSPSR .................................. 3056
TPIU_PIDR0 ................................. 3063
TPIU_FFSR .................................. 3058
TPIU_PIDR0 .................................. 3062
TPIU_SSPR .................................. 3057
TPIU_SSPSR .................................. 3056

TIMx_CNT ............................... 1545, 1667, 1703, 1765, 1860
TIMx_CCR6 ..................... 1550, 1665, 1764, 1857
TIMx_CCER .............. 1541, 1550, 1665, 1764, 1857
TIMx_CCMR1 .... 1532, 1534, 1569, 1661, 1761-1762
TIMx_CCMR2 .... 1537-1538, 1663-1664
TIMx_CCMR3 .......................... 1556
TIMx_CCR1 .... 1547, 1669-1670, 1767, 1862
TIMx_CCR2 ........ 1547, 1671-1672
TIMx_CCR3 .... 1548, 1673-1674
TIMx_CCR4 .... 1549, 1675-1676
TIMx_CCR5 .......................... 1554
TIMx_CCR6 .......................... 1555
TIMx_CNT .... 1545, 1667, 1703, 1765, 1860
TIMx_CR1 .... 1518, 1648, 1700, 1758, 1849
TIMx_CR2 .... 1519, 1649, 1702, 1855
TIMx_DCR .... 1565, 1681, 1870
TIMx_DIER .... 1527, 1655, 1702, 1759, 1851
TIMx_DMAR .... 1567, 1682, 1871
TIMx_DTR2 ........ 1557, 1866
TIMx_EGR .... 1531, 1658, 1703, 1760, 1853
TIMx_PSC .... 1545, 1668, 1704, 1766, 1860
TIMx_RCR .... 1546, 1861
TIMx_SMCR .... 1523, 1651
TIMx_SR .... 1528, 1656, 1703, 1759, 1852
TIMx_TISEL ... 1559, 1678, 1767, 1867
TPIU_ACPR .................................. 3057
TPIU_CIDR0 .......................... 3064
TPIU_CIDR1 .................................. 3065
TPIU_CIDR2 .................................. 3065
TPIU_CIDR3 .................................. 3066
TPIU_CLAIMCLR .................................. 3060
TPIU_CLAIMSETR .................................. 3060
TPIU_CPSR .................................. 3056
TPIU_DEVIDR .................................. 3061
TPIU_DEVTYPEP .................................. 3061
TPIU_FFCR .................................. 3058
TPIU_FFISR .................................. 3058
TPIU_PIDR1 .................................. 3063
TPIU_PIDR2 .................................. 3063
TPIU_PIDR3 .................................. 3064
TPIU_PIDR4 .................................. 3062
TPIU_PSCR .................................. 3059
TPIU_PSS .................................. 3057
TPIU_PIU .................................. 3062

U

UCPD_CFGCR1 .................................. 2647
UCPD_CFGCR2 .................................. 2649
UCPD_CFGCR3 .................................. 2650
UCPD_CR .................................. 2650
UCPD_ICR .................................. 2657
UCPD_IMR .................................. 2653
UCPD_RX_ORDEXTR1 .................................. 2662
UCPD_RX_ORDEXTR2 .................................. 2662
UCPD_RX_ORDSETR .................................. 2660
UCPD_RX_PAYSZR .................................. 2661
UCPD_RXDR .................................. 2661
UCPD_SR .................................. 2654
UCPD_TX_ORDSETR .................................. 2658
UCPD_TX_PAYSZR .................................. 2659
UCPD_TXDR .................................. 2659
USART_BRR .................................. 2296
USART_CR1 .................................. 2280, 2284
USART_CR2 .................................. 2287
USART_CR3 .................................. 2291
USART_GTPR .................................. 2296
USART_ICR .................................. 2310
USART_ISR .................................. 2299, 2305
USART_POST .................................. 2313
USART_RDR .................................. 2312
USART_RQR .................................. 2298
USART_RTOR .................................. 2297
USART_TDR .................................. 2312
USB_BCR .................................. 2610
USB_CEP_RXTBX .................................. 2623
USB_CEP_TXTBX .................................. 2622
USB_CEP_TXRX .................................. 2620
USB_CEP_TXRX .................................. 2622
USB_CEP_RX .................................. 2611
USB_CNTR .................................. 2601
USB_DADDR .................................. 2608
USB_FNR .................................. 2608
USBISTR .................................. 2604
USBLPMSR .................................. 2609

V

VREFBUF_CCR .................................. 1194
VREFBUF_CSR .................................. 1193

3117/3119
Index

W
WWDG_CFR .......................... 1945
WWDG_CR .......................... 1944
WWDG_SR .......................... 1946
IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2023 STMicroelectronics – All rights reserved